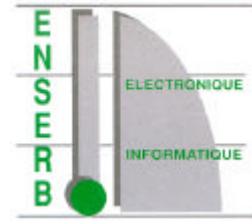




I.U.T. – Université BORDEAUX 1
Département Génie Electrique
et Informatique Industrielle
33405 TALENCE CEDEX



Ecole Nationale Supérieure
d'Electronique
et de Radioélectricité
de BORDEAUX
33402 TALENCE CEDEX

RAPPORT DE STAGE

LES MICROCONTROLEURS PIC

PRINCIPE ET MISE EN ŒUVRE

APPLICATION : CODAGE DE VOIX ADPCM

Vincent ROUGEOT

Maître de stage :
Maître de conférence, Monsieur KADIONIK

Enseignant responsable :
Professeur agrégé, Madame VELIZ

Année Universitaire 1999 - 2000

LES MICROCONTROLEURS PIC
PRINCIPE ET MISE EN ŒUVRE
APPLICATION : CODAGE DE VOIX ADPCM

Vincent ROUGEOT

Remerciements

Je tiens à remercier très sincèrement Monsieur KADIONIK pour son accueil à l'ENSERB et de m'avoir donné, durant ce stage, ses conseils pédagogiques et apporté son aide scientifique et technique indispensable à son excellent déroulement. Je le remercie aussi pour la grande et confiante autonomie qu'il a bien voulu me laisser dans la conduite de mes recherches et l'exécution de mes travaux.

J'adresse également tous mes remerciements aux enseignants et aux personnels de l'Ecole qui m'ont aidé et offert d'excellentes conditions pour mener à bien mon projet.

Résumé

Ce rapport de stage, regroupe un ensemble d'informations théoriques et pratiques qui permettent de découvrir et d'utiliser les microcontrôleurs PIC de la société américaine Microchip. Dans ce dossier, nous aborderons, dans un premier temps, les caractéristiques spécifiques des microcontrôleurs PIC, puis, nous verrons les systèmes nécessaires à leur développement, tels que les programmeurs et logiciels de programmation. Nous terminerons par l'étude et la réalisation d'une application traitant du codage de voix, par algorithme ADPCM.

Toutes ces informations seront détaillées et commentées par des schémas, figures et impressions d'écran. La documentation technique et les logiciels utilisés, durant ce stage, seront disponibles sur un CD-ROM joint à ce rapport.

Ce document pourra éventuellement servir de support à l'élaboration de cours ou travaux pratiques destinés aux élèves ingénieurs de l'ENSERB ou d'étudiant de l'IUT.

Abstract

This placement report composes a set of theoretical and practical information which allow to discover and use Microchip's PIC microcontrollers. First, we'll see specific characteristics of PIC microcontrollers and then we'll see the systems necessary to develop them, like programmers and programming softwares. We'll finish by the study and the realization of an application using ADPCM algorithm. All these information will be detailed and commented by diagrams, pictures and nag screens.

Technical documents and softwares, used during this placement, will be available on a CD-ROM joined to this report.

This book will possibly be the base of lessons or practical works intended to ENSERB engineers or IUT students.

Mots clés

Microcontrôleurs PIC – Programmeur de PIC – Assembleur – Compilateur C
Compression – Décompression – Codage de voix – ADPCM

Remerciements	3
Résumé	4
Abstract	4
Mots clés	4
Sommaire	5
I – Présentation de l'ENSERB	7
II – Les microcontrôleurs PIC	8
II.1 – Principe de l'architecture des microcontrôleurs.....	8
II.2 – Architecture Harvard.....	9
II.3 – Les cycles machine.....	9
II.4 – Circuit RISC.....	10
II.5 – Organisation mémoire.....	13
II.5.1 – Organisation de la mémoire de programmes	13
II.5.2 – Organisation de la mémoire de données	14
II.6 – Circuits périphériques.....	14
II.7 – Schéma bloc d'un microcontrôleur PIC	15
II.8 – Les familles PIC.....	16
III- Système de développement pour microcontrôleurs PIC	17
III.1 – Base d'un système de développement.....	17
III.1.1 – Côté logiciel.....	17
III.1.2 – Côté matériel.....	17
III.2 – MPLAB de Microchip.....	18
IV – Etude et réalisation d'un programmeur pour PIC 16F84	22
IV.1 – Etude et réalisation du programmeur P ARPIC	22
IV.1.1 – Etude du programmeur P ARPIC de David Tait.....	22
IV.1.2 – Réalisation du programmeur P ARPIC	25
IV.1.3 – Fonctionnement du programmeur P ARPIC	27
IV.2 – Etude et réalisation du programmeur QUICK AND DIRTY.....	31
IV.2.1 – Etude du programmeur QUICK AND DIRTY de David Tait	31
IV.2.2 – Réalisation du programmeur QUICK AND DIRTY	32
IV.2.3 – Fonctionnement du programmeur QUICK AND DIRTY	33
IV.3 – Etude et réalisation du programmeur PICPROG 2000.....	35
IV.3.1 – Etude du programmeur PICPROG 2000 de Jacques Weiss	35
IV.3.2 – Réalisation du programmeur PICPROG 2000.....	36
IV.3.3 – Fonctionnement du programmeur PICPROG 2000	37

V – Application : codage de voix ADPCM	39
V.1 – Origine de cette application	39
V.2 – Cahier des charges.....	39
V.3 – Etude théorique de l'application.....	39
V.4 – Etude matérielle	41
V.4.1 – Etude du préamplificateur	42
V.4.2 – Etude du filtre passe bas d'entrée	43
V.4.3 – Etude du Convertisseur Analogique Numérique (CAN)	44
V.4.4 – Etude du compteur d'adresses	45
V.4.5 – Etude du module PWM.....	46
V.4.6 – Etude de l'amplificateur audio	49
V.4.7 – Etude de l'alimentation.....	50
V.4.8 – Schéma électrique complet du circuit.....	51
V.5 – Réalisation du circuit.....	52
V.6 – Le codage ADPCM.....	54
V.5.1 – Quel algorithme utiliser ?.....	54
V.5.2 – Compression / Décompression.....	54
V.5.3 – Au niveau informatique.....	55
V.7 – Etude logicielle	56
V.7.1 – Choix du langage de programmation.....	56
V.7.2 – Choix du compilateur C.....	56
V.7.3 – Fonctionnement du compilateur HI-TECH C Compiler	57
V.8 – Développement du programme du circuit ADPCM.....	61
 Conclusion.....	 63
 Bibliographie	 64
 Annexes	 65

I – Présentation de l'ENSERB

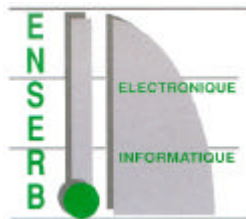
Née en 1920 au sein de la Faculté des Sciences de Bordeaux, l'ENSERB est l'une des plus anciennes Ecoles d'Ingénieurs dans le domaine de l'électronique. Elle est habilitée à délivrer le titre d'ingénieur diplômé en 1934 et prend le statut d'Ecole Nationale Supérieure d'Ingénieurs en 1975. En 1982, elle est habilitée à délivrer le diplôme d'ingénieur par la voie de la formation continue. Une seconde filière d'ingénieurs dans le domaine de l'informatique est ouverte en 1986.

Actuellement, l'ENSERB possède deux filières de formation :

- La filière Electronique avec les options Automatique - Robotique, Traitement du Signal, Informatique Industrielle, Micro-électronique et Télécommunications.
- La filière Informatique avec les options Génie Logiciel, Calcul Parallèle et Distribué, Réseaux et Systèmes Répartis, Technologies de l'Image et de la Communication.

Pour répondre à ses missions : formation initiale et continue, recherche et transferts de technologies, l'ENSERB s'appuie sur des laboratoires de recherche de niveau international, tel que le laboratoire de Micro-électronique IXL, le Laboratoire Bordelais de Recherche Informatique (LaBRI), le Laboratoire d'Automatique et de Productique (LAP) et l'Equipe Signal et Image (ESI), qui garantissent la compétence de ses enseignants, favorisent ses relations avec les entreprises, et facilitent son insertion dans des réseaux d'échanges internationaux.

L'école est implantée au cœur du Campus Universitaire de Talence - Pessac - Gradignan, fréquenté par plus de 50 000 Etudiants, à proximité immédiate des Laboratoires, d'une Bibliothèque Universitaire, d'un village universitaire et d'un Restaurant Universitaire.



Ecole Nationale Supérieure d'Electronique et de Radioelectricité de Bordeaux
Avenue du Docteur Schweitzer
Domaine Universitaire
BP 99 – 33402 Talence Cedex
Tél : 05 56 84 65 00 Fax : 05 56 37 20 23
<http://www.enserb.fr>

II – Les microcontrôleurs PIC

Quasiment inconnus il y a seulement cinq ans, les microcontrôleurs PIC de Microchip sont aujourd'hui omniprésents dans les appareils les plus divers : programmeurs domestiques, télécommandes, appareils électroménagers, etc.

La raison de leur succès tient tout à la fois au coût particulièrement bas des circuits et à leurs performances remarquables. Leur jeu d'instructions réduit, plus connu sous l'acronyme de RISC, et leur architecture de type Harvard y sont aussi pour beaucoup.

II.1 – Principe de l'architecture des microcontrôleurs

Sur tous les systèmes microprocesseurs/microcontrôleurs nous trouvons des modules indispensables au fonctionnement du processeur :

- le bus (de données, d'adresses, de commandes)
- l'unité centrale (CPU)
- la mémoire (EPROM, RAM)
- les entrées/sorties (I/O)

Les bus permettent au processeur de communiquer avec les autres composants. Cette communication pouvant être série ou parallèle.

L'unité centrale exécute les instructions et décide comment le programme doit fonctionner.

La mémoire est utilisée pour stocker des instructions (le programme) et des données.

Les entrées/sorties sont utilisées par le processeur pour communiquer avec le monde extérieur.

Comme nous le verrons par la suite, il existe deux types d'architectures différentes utilisant ces modules.

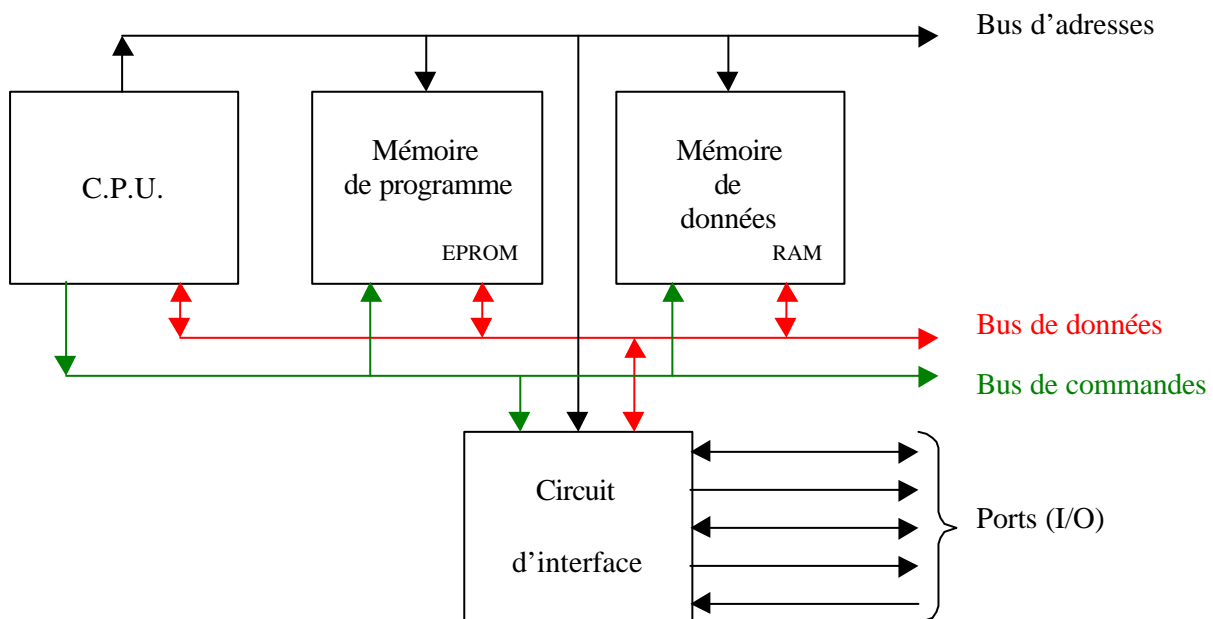


Figure 1 : Constitution d'un système à microprocesseur / microcontrôleur.

II.2 – Architecture Harvard

Presque tous les microcontrôleurs actuels utilisent une architecture interne dite de von-Neuman, architecture commune à celle que nous rencontrons dans la plupart des ordinateurs.

La mémoire dite de programme contient aussi bien des instructions que des données et nous ne disposons que d'un bus, appelé bus de données, qui véhicule tour à tour les codes des instructions et les données.

Si cette architecture est très performante, elle pose certains problèmes lorsque nous voulons faire fonctionner l'ensemble rapidement. Il est alors préférable de faire appel à une structure dite Harvard dans laquelle les instructions et les données sont différenciées et véhiculées sur des bus différents. Ceci permet d'augmenter la vitesse d'exécution des programmes.

La figure suivante montre les deux types d'architecture.

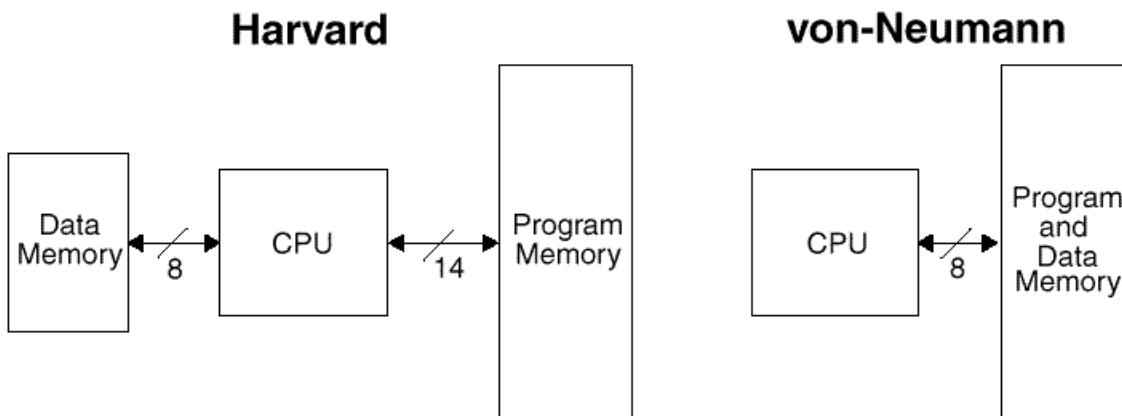


Figure 2 : Architecture Harvard et von-Neuman.

Les microcontrôleurs PIC utilisent donc une architecture Harvard mais font aussi appel à une architecture de type RISC contrôlée par une horloge interne spéciale : les cycles machine.

II.3 – Les cycles machine

L'horloge externe d'un microcontrôleur est divisée par 4 en interne pour créer quatre horloges nommées Q1, Q2, Q3 et Q4. Les instructions sont alors traitées avec une fréquence qui est le quart de la fréquence de l'horloge d'entrée.

L'addition des durées de chaque Qx représente un cycle machine. C'est pendant ce temps que l'instruction est décodée et exécutée. En interne, le pointeur d'instruction est incrémenté d'une unité à chaque cycle Q1 alors que les instructions sont recherchées en mémoire de programme et mémorisées dans le registre d'instruction à chaque cycle Q4.

Le diagramme suivant met clairement en évidence le fait qu'une instruction s'exécute alors que la suivante est en cours de recherche.

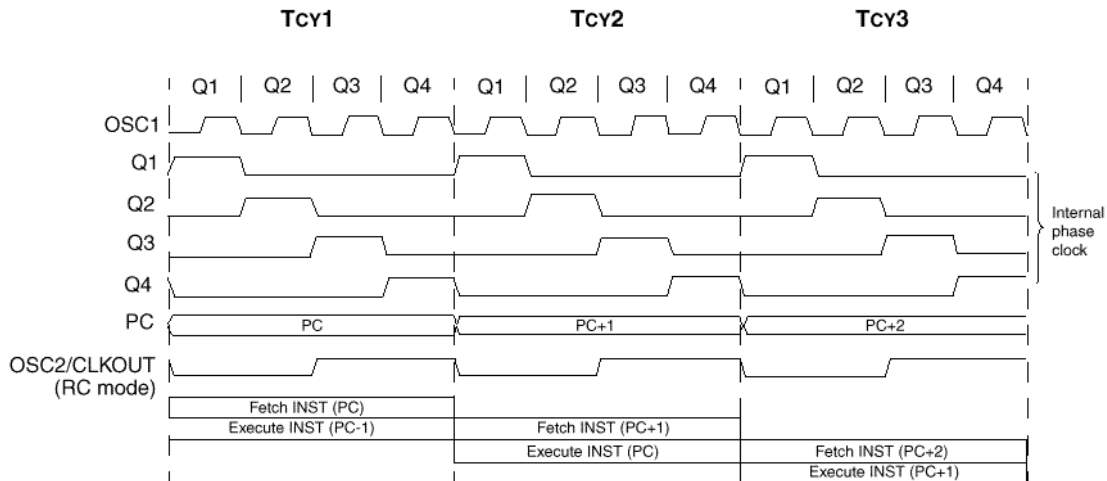


Figure 3 : Les cycles machine.

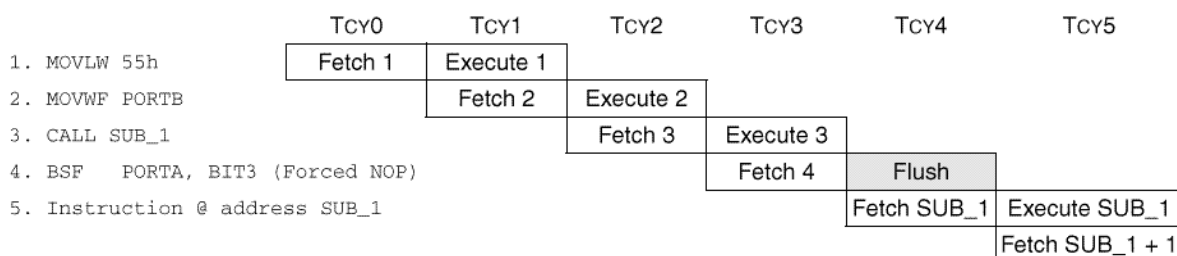
- * Fetch = rechercher
- * Execute = exécuter

II.4 – Circuit RISC

Comme nous l'avons vu précédemment, les microcontrôleurs PIC utilisent une architecture de type Harvard comportant un circuit de type RISC. RISC signifie Reduced Instruction Set Computer qui se traduit par : circuit à jeu d'instructions réduit. Un circuit de type RISC dispose d'une structure de pipeline qui lui permet au minimum d'exécuter une instruction pendant qu'il est entrain de rechercher la suivante.

Ce principe permet d'accroître la vitesse d'exécution des programmes par rapport aux microprocesseurs classiques, appelés par opposition de type CISC, ce qui veut dire Complex Instruction Set Computer.

La figure 4 montre le fonctionnement de pipeline, Tcyx représentant un cycle machine.



- * Fetch = rechercher
- * Execute = exécuter
- * Flush = vider

Figure 4 : Principe de pipeline.

Explication de l'exemple de la figure 4 :

Cycle n°0 : Une nouvelle instruction est lue (MOVLW 55h).

Cycle n°1 : Une nouvelle instruction est lue (MOVWF PORTB).
L'instruction lue au cycle n°0 est exécutée.
Le registre W reçoit 55h.

Cycle n°2 : Une nouvelle instruction est lue (CALL SUB_1).
L'instruction lue au cycle n°1 est exécutée.
Le registre W est recopié dans le PORTB.

Cycle n°3 : Une nouvelle instruction est lue (BSF PORTA, BIT3).
L'instruction lue au cycle n°2 est exécutée.
Le pointeur d'instruction pointe sur l'adresse de SUB_1.

Cycle n°4 : La première instruction de SUB_1 est lue.
Aucune instruction n'est exécutée.
L'information lue au cycle n°3 est supprimée.
Le temps d'exécution du CALL est augmenté d'un cycle.

Cycle n°5 : Une nouvelle instruction est lue à l'adresse de SUB_1 + 1.
L'instruction lue au cycle n°4 est exécutée.

De plus, un vrai circuit RISC doit en principe exécuter toutes les instructions à la même vitesse, c'est à dire en un cycle machine. Cependant, il existe quelques exceptions où certaines instructions durent 2 cycles comme l'instruction CALL de l'exemple précédent.

Les PIC sont des circuits RISC qui comportent seulement 30 ou 35 instructions. Malgré ce jeu d'instruction réduit, les microcontrôleurs PIC sont quasiment aussi performants, en termes de compacité de code, que le jeu plus complet des microcontrôleurs conventionnels.

La figure suivante présente le jeu d'instruction des microcontrôleurs PIC.

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

Figure 5 : Jeu d'instructions des microcontrôleurs PIC.

II.5 – Organisation mémoire

Comme nous l'avons vu précédemment, une architecture Harvard dispose de deux zones mémoire : une zone mémoire de donnée et une zone mémoire de programme. Lesquelles disposent chacune de leur propre bus.

II.5.1 – Organisation de la mémoire de programmes

Le propre des instructions RISC est de regrouper en seul mot le code de l'instruction et l'opérande ou son adresse. La mémoire de programme est alors organisée en mots de 12, 14 ou 16 bits suivant les familles des différents microcontrôleurs. Cette mémoire est découpée en pages dont la taille est fonction de la totalité de la mémoire de programme. La mémoire de programme possède aussi une pile à plusieurs niveaux. La mémoire de programme peut être une ROM, une EPROM ou une EEPROM.

La figure 6 montre l'organisation de la mémoire de programme.

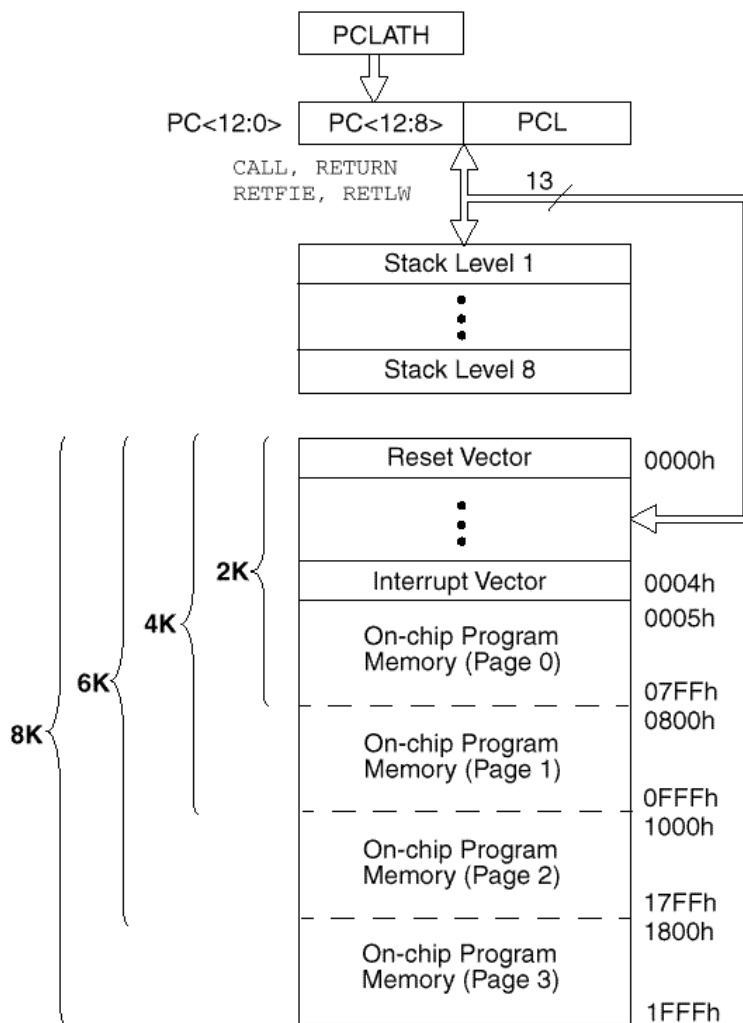


Figure 6 : Organisation de la mémoire de programme.

II.5.2 – Organisation de la mémoire de données

Bien que les circuits fonctionnent avec des instructions sur 12, 14 ou 16 bits, ils utilisent des données codées sur 8 bits. Ces données se trouvent dans la RAM interne du circuit.

Une partie de cette RAM contient des registres à fonction dédiée (Special Function Registers) qui servent à initialiser et à piloter les microcontrôleurs (TRISA, TRISB, STATUS etc.). Ces registres ont des adresses réservées et ne peuvent être utilisés pour les variables de l'utilisateur.

L'autre partie de la RAM, (General Purpose Register) ne contenant pas de SFR, est utilisée pour enregistrer des variables.

La figure ci-dessous montre un exemple d'organisation de la mémoire de données.

	File Address		File Address
INDF	00h	INDF	80h
TMR0	01h	OPTION_REG	81h
PCL	02h	PCL	82h
STATUS	03h	STATUS	83h
FSR	04h	FSR	84h
PORTA	05h	TRISA	85h
PORTB	06h	TRISB	86h
	07h	PCON	87h
ADCON0 / EEDATA ⁽²⁾	08h	ADCON1 / EECON1 ⁽²⁾	88h
ADRES / EEADR ⁽²⁾	09h	ADRES / EECON2 ⁽²⁾	89h
PCLATH	0Ah	PCLATH	8Ah
INTCON	0Bh	INTCON	8Bh
General Purpose Registers ⁽³⁾	0Ch	General Purpose Registers ⁽⁴⁾	8Ch
	7Fh		FFh
Bank0		Bank1	

Figure 7 : Organisation de la mémoire de données.

II.6 – Circuits périphériques

Outre les ports d'entrées/sorties que possèdent tous les microcontrôleurs, des membres de certaines familles de PIC possèdent des circuits périphériques : convertisseur analogique/numérique, circuits de capture compare, timers, ports série (SPI, I²C, SCI), etc.

II.7 – Schéma bloc d'un microcontrôleur PIC

La figure 8, ci-dessous, représente l'architecture interne d'un microcontrôleur PIC de la famille Mid-range. Nous pouvons voir qu'il est composé de 7 ports d'entrées/sorties, d'un convertisseur analogique/numérique, de 2 ports série, d'un driver d'afficheur LCD, etc.

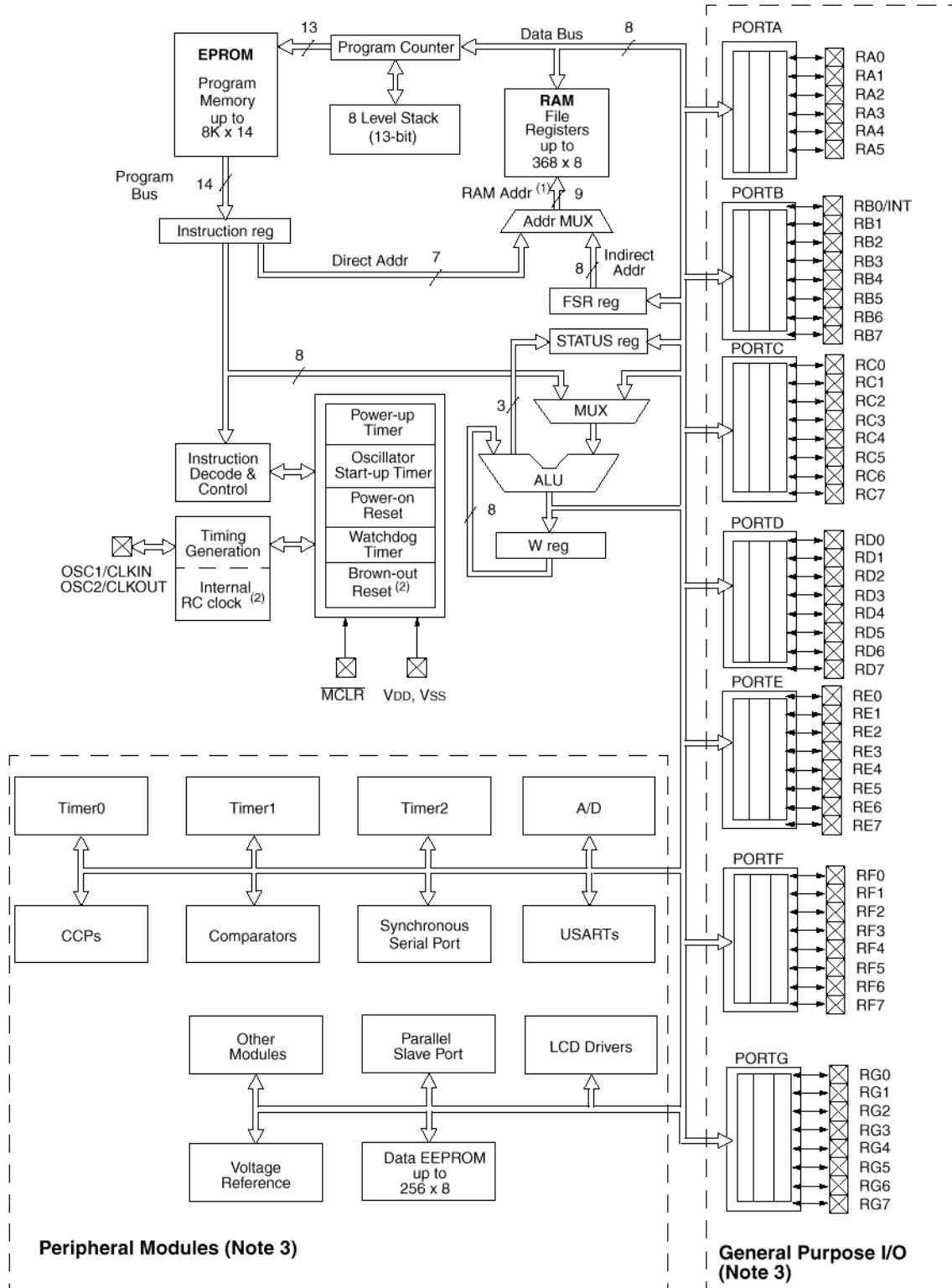


Figure 8 : Schéma bloc d'un microcontrôleur PIC.

II.8 – Les familles PIC

Les différents microcontrôleurs sont regroupés en fonction de la taille de leurs instructions :

- Base-line : instructions sur 12 bits
- Mid-range : instructions sur 14 bits
- High-end : instructions sur 16 bits

La famille la plus connue est sans conteste la famille 16C5X car la plus ancienne mais c'est aussi celle que l'on rencontre dans la plupart des applications. Elle fait partie de la famille Base-line puisque ses instructions sont codées sur 12 bits.

La deuxième famille qualifiée par Microchip de Mid-range comprend actuellement quatre références mais est appelée à se développer rapidement. Ces circuits 16C7XX reprennent les mêmes caractéristiques que celles de la famille 16C5X mais disposent de ressources internes plus nombreuses avec des ports série variés ou bien encore un convertisseur analogique/numérique.

La troisième famille, basée sur une architecture 16 bits, concerne les 17CXX. C'est la famille haut de gamme des microcontrôleurs 8 bits.

De plus, tous ces circuits se distinguent tout à la fois par la taille et le type de leur mémoire, par le nombre de ports d'entrées/sorties et par les différents circuits périphériques qu'ils comportent.

	Fréquence maximale en MHz	Taille de l'EPROM	Taille de la ROM	Taille de l'EEPROM	Taille de la RAM en octets	Timers	Capture Comparaison PWM	Types de ports série (SPI / I ² C / SCI)	Nombre de voix CAN	Nombre de sources d'interruptions	Nombre d'entrées / sorties	Nombre d'instructions	Boîtiers
12C508	4	512 x12	-	-	25	WDT TMR0	-	SPI	-	-	6	33	8 DIP 8 SOIC
12C671	10	1K x12	-	-	128	WDT TMR0	-	SPI	4	3	6	35	8 DIP 8 SPOIC
16C55	20	1K x12	-	-	24	WDT TMR0	-	-	-	-	12	33	18 DIP 18 SOIC 20 SSOP
16C57	20	2K x12	-	-	72	WDT TMR0	-	-	-	-	20	33	28 DIP 28 SOIC
16CR58A	20	-	2K x12	-	73	WDT TMR0	-	-	-	-	12	33	18 DIP 18 SOIC 20 SSOP
16C72	20	2K x14	-	-	128	WDT TMR0 TMR1 TMR2	1	SPI I ² C SCI	5	8	22	35	28 DIP 28 SOIC 28 SSOP
16F84	10	-	-	1K x14	36	WDT TMR0	-	SPI	-	4	13	35	18 DIP 18 SOIC
17C42	33	2K x16	-	-	238	WDT TMR0 TMR1 TMR2 TMR3	2	SPI I ² C SCI	-	11	33	58	40 DIP 44 PLCC 44 QFP
17CR43	33	-	4K x16	-	454	WDT TMR0 TMR1 TMR2 TMR3	2	SPI I ² C SCI	-	11	33	58	40 DIP 44 PLCC 44 QFP

Figure 9 : Caractéristique de quelques circuits PIC.

III- Système de développement pour microcontrôleurs PIC

La mise en œuvre d'un système à microprocesseur ou microcontrôleur requiert généralement des outils bien précis. Sur le plan matériel, il est indispensable de disposer de programmeurs pour implanter les programmes dans la mémoire des microcontrôleurs. Du côté logiciel, il est nécessaire d'avoir une chaîne de compilation.

III.1 – Base d'un système de développement

III.1.1 – Côté logiciel

Un système de développement comporte en premier lieu un assembleur et parfois un ou des compilateurs adaptés au langage évolué que l'on souhaite utiliser pour programmer.

L'assembleur traduit les instructions écrites en utilisant les mnémoniques du langage machine en code binaire exécutable par le microcontrôleur.

Le compilateur quant à lui traduit les instructions écrites en langage évolué (BASIC, C, PASCAL) qui constituent aussi ce que nous appelons le listing ou code source, en code binaire exécutable par le microcontrôleur qui constitue le code objet.

Dans un système de développement bien conçu, les deux programmes, assembleur et compilateur, peuvent coexister et être utilisables l'un et l'autre sans difficulté. Ces deux programmes, assembleur et/ou compilateur doivent nécessairement fonctionner sur une machine appelée machine hôte. Cette machine peut être : un système spécifique du fabricant des microcontrôleurs, une station de travail, un calculateur ou un PC.

Une fois le programme de l'application écrit et assemblé ou compilé sur la machine hôte, nous sommes en possession d'un binaire exécutable.

III.1.2 – Côté matériel

Pour implanter le binaire exécutable, nous avons besoin d'un programmeur. Cet appareil est en fait un système qui va transférer de la machine hôte, le code objet du programme dans la mémoire du microcontrôleur.

Le chapitre IV de ce rapport nous montre la réalisation de quelques programmeurs pour microcontrôleurs PIC 16F84.

III.2 – MPLAB de Microchip

Pour développer des applications en assembleur, plusieurs programmes assembleur existent sur le marché. Cependant le choix, d'un tel logiciel, s'est porté sur l'outil MPLAB 5.0 mis à disposition gratuitement sur Internet par la société Microchip. Ce logiciel fonctionne sur PC, sous le système d'exploitation Windows95/98 de Microsoft.

La suite logicielle comprend tout ce qu'il faut pour assembler les programmes : éditeur de texte, assembleur, simulateur, éditeur de registres, etc. Il supporte aussi plusieurs émulateurs.

L'interface graphique de ce logiciel est représentée en figure 10.

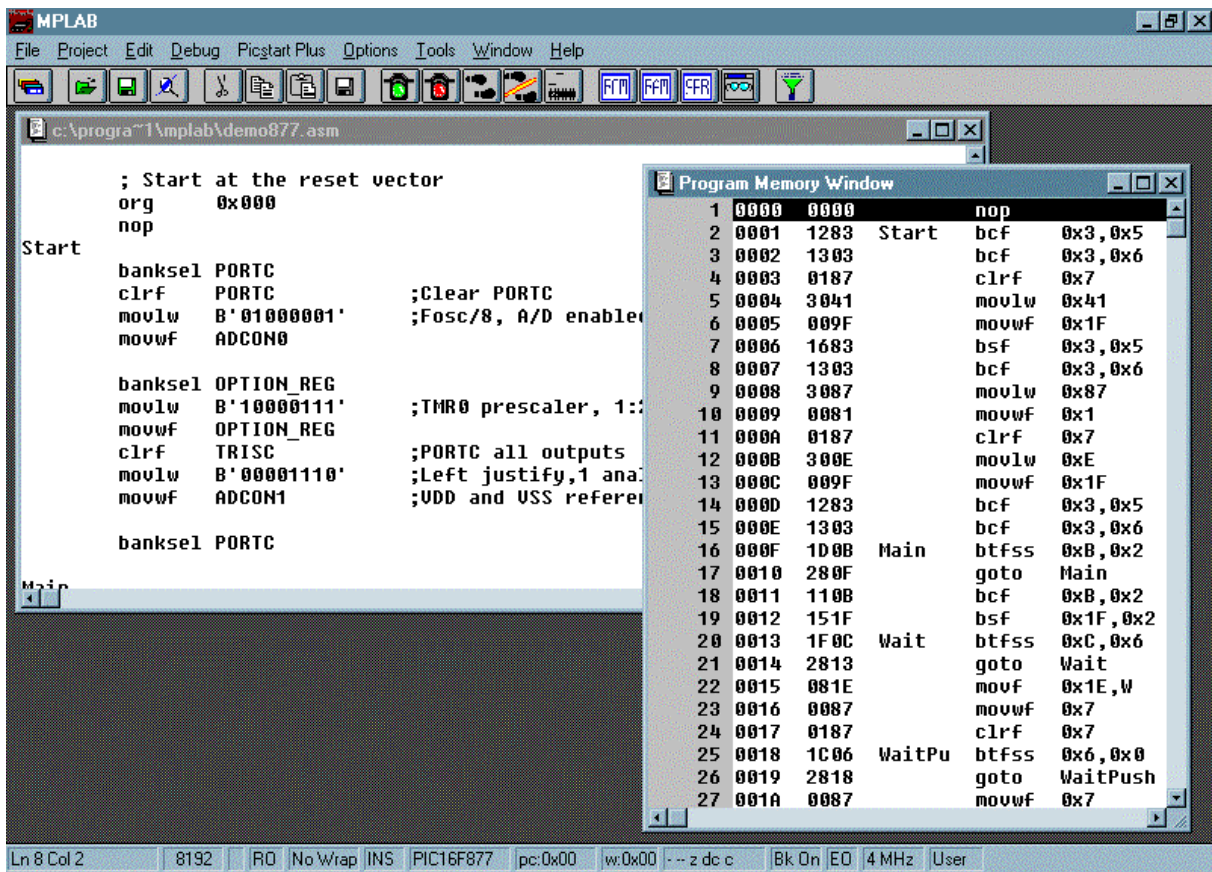
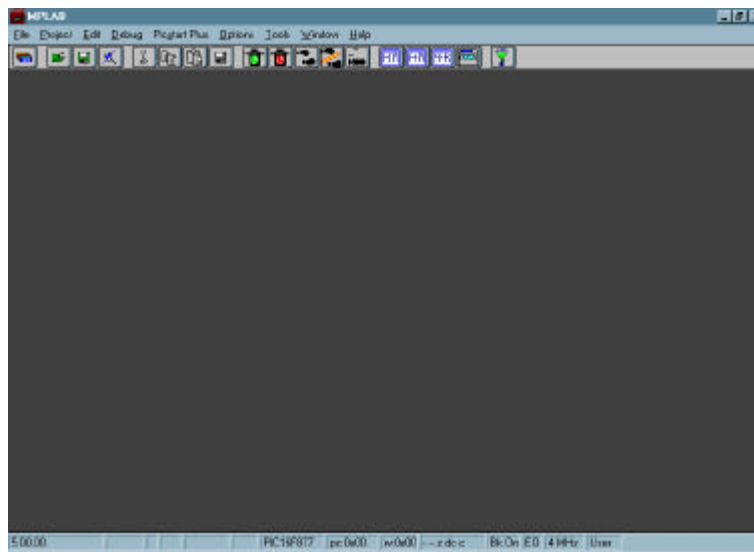


Figure 10 : Interface graphique du logiciel MPLAB.

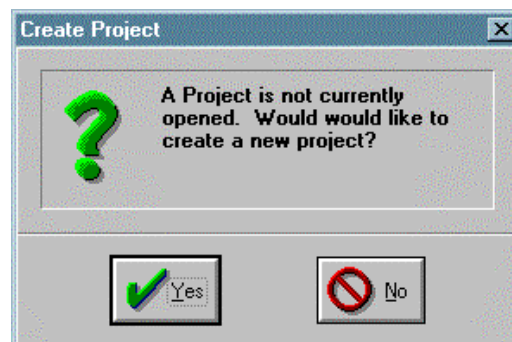
- ✓ Ce logiciel est disponible sur le CD-ROM :
Répertoire : Mplab50
Référence : mplab50.exe

Voyons maintenant la marche à suivre pour créer un premier programme et son code binaire exécutable.

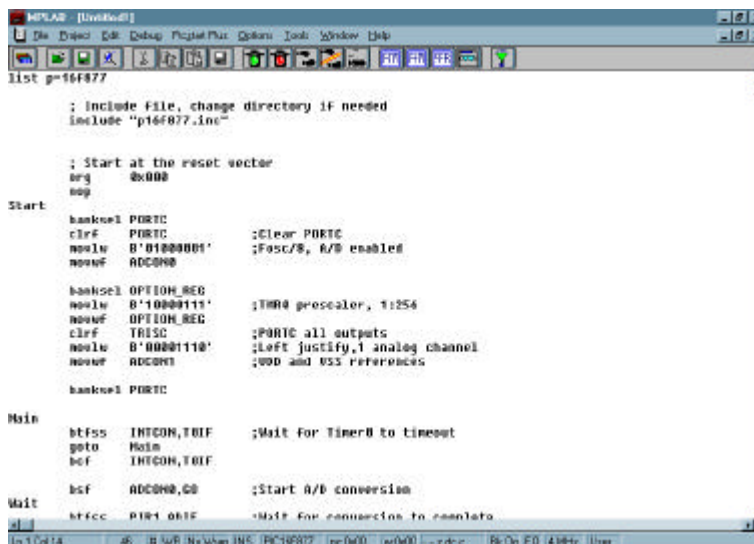
Après avoir lancé le logiciel MPLAB, la fenêtre suivante apparaît.



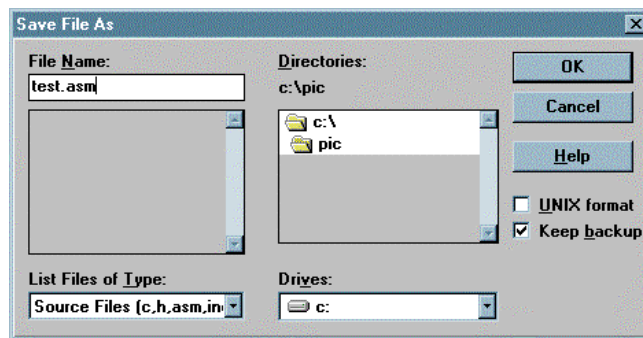
- Cliquer sur **File** puis **New**, une fenêtre d'édition de texte apparaît, ainsi qu'un message demandant si l'utilisateur souhaite créer un nouveau projet. Répondre par **NO**.



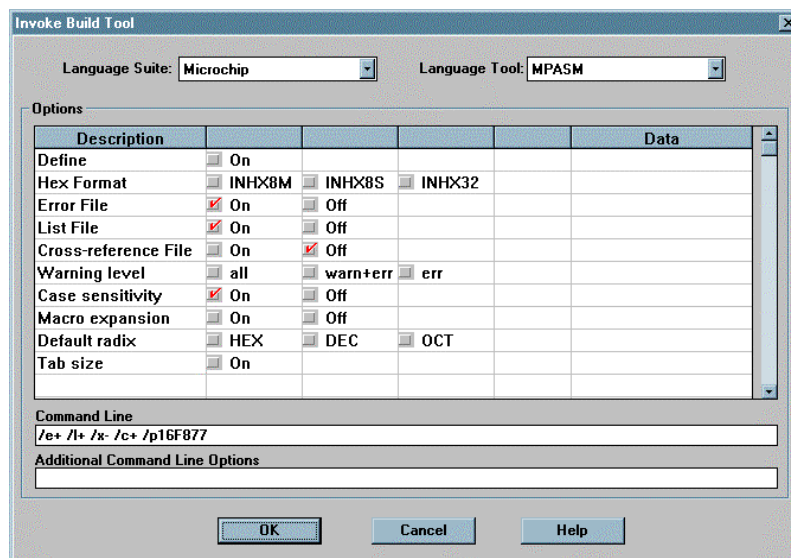
- Le message disparaît.
- Dans la fenêtre d'édition de texte, taper le code source du programme.



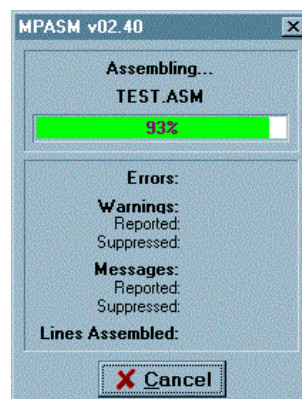
- Enregistrer le fichier en cliquant sur **File** puis **Save As...** . La fenêtre **Save File As** apparaît.



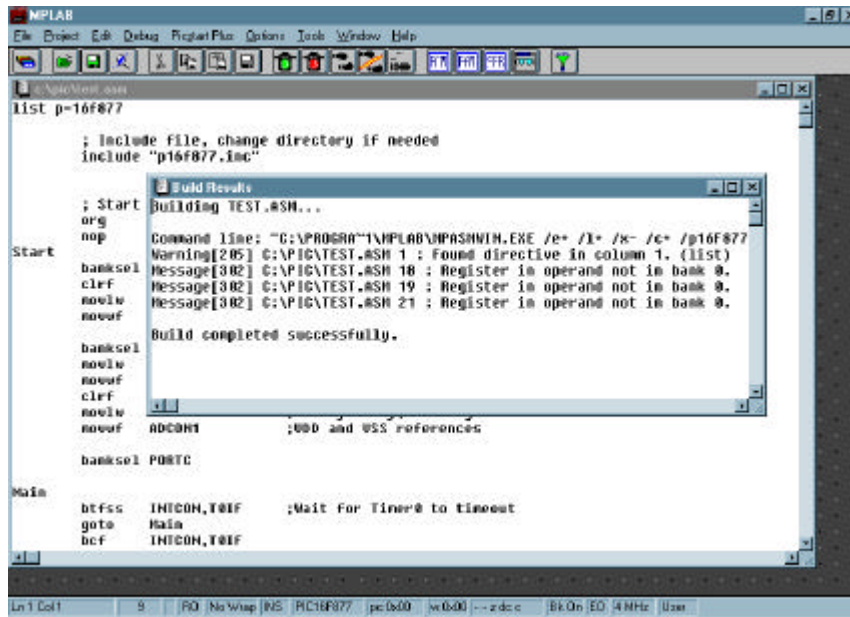
- Dans le champs **File Name:**, taper le nom du fichier (ici **test.asm**). Dans le champs **Directories:**, choisir l'emplacement où sera enregistré le fichier (ici **c:\pic**), puis cliquer sur **OK**.
- La fenêtre **Save File As** se ferme.
- Pour assembler le programme, cliquer sur **Project** puis **Build Node** . La fenêtre **Invoke Build Tool** apparaît.



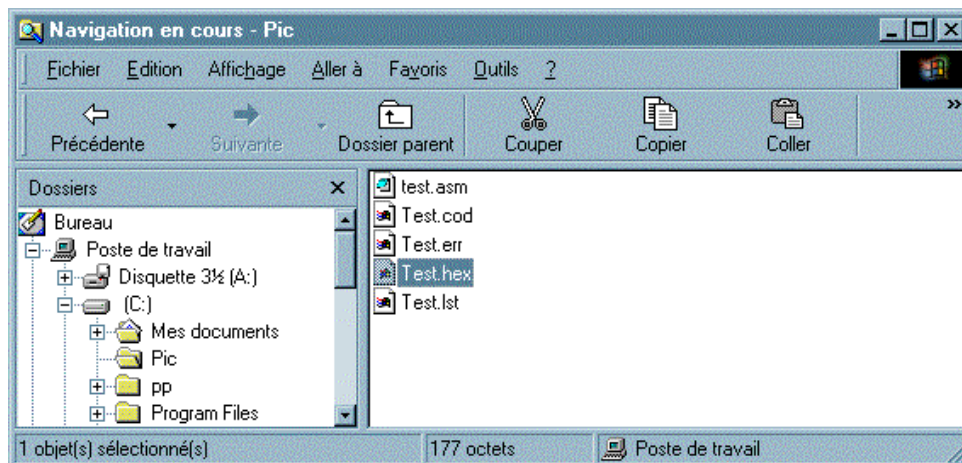
- Modifier les options si besoin, puis cliquer sur **OK**.
- La compilation du fichier **test.asm** commence.



- Lorsque la compilation est terminée, la fenêtre **Build Results** apparaît et informe de l'état de la compilation.



- Vérifier que le message **Buil completed successfully** y figure. Dans le cas contraire, une ou plusieurs erreurs figurent dans le code source.
- Le code binaire, **test.hex**, est alors disponible dans le dossier **c:\pic** où a été enregistré **test.asm**



IV – Etude et réalisation d'un programmeur pour PIC 16F84

Le choix d'un PIC 16F84 s'explique par le fait qu'il s'agit d'un microcontrôleur possédant une mémoire programme EEPROM, c'est à dire : programmable et effaçable électriquement. Ce qui lui confère une grande souplesse d'utilisation.

- ✓ *La documentation technique du PIC 16F84 est disponible sur le CD-ROM :*
Répertoire : Doc
Référence : PIC16F84.pdf

La programmation des microcontrôleurs PIC se fait par l'intermédiaire d'un ordinateur et d'une suite de logiciels.

IV.1 – Etude et réalisation du programmeur PARIPI

IV.1.1 – Etude du programmeur PARIPI de David Tait

L'étude du programmeur repose sur les plans de circuits réalisés par David Tait, un ingénieur anglais.

Celui-ci a réalisé différents circuits dont plusieurs programmeurs, interfaces séries, interfaces parallèles et cartes de tests.

Pour programmer le PIC16F84, il a été décidé d'utiliser le port parallèle d'un PC et d'ajouter au programmeur un circuit de test afin de vérifier l'intégrité de la programmation.

Le schéma bloc de la figure 11 représente le système à réaliser.

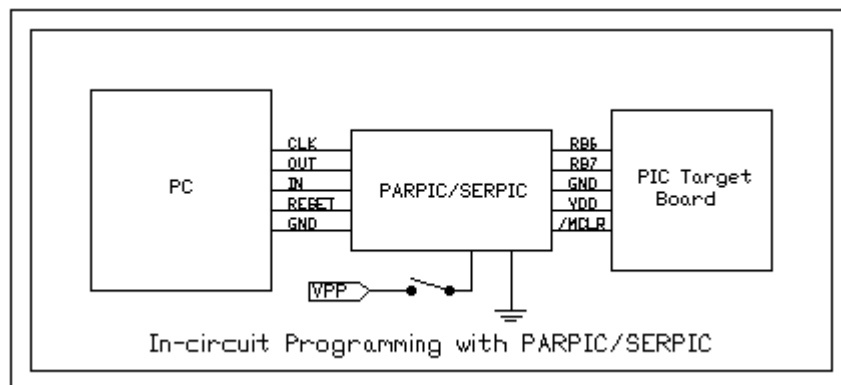


Figure 11 : Schéma bloc du programmeur PARIPI.

Pour réaliser ce programmeur, deux circuits ont été retenus :

- L'interface parallèle est réalisée à partir de PARIPI INTERFACE de David Tait.
- Le circuit de test est réalisé à partir de RE-PROGRAMMABLE TEST CIRCUIT de David Tait.

Les schémas de ces deux circuits sont représentés en figure 12 et figure 13.

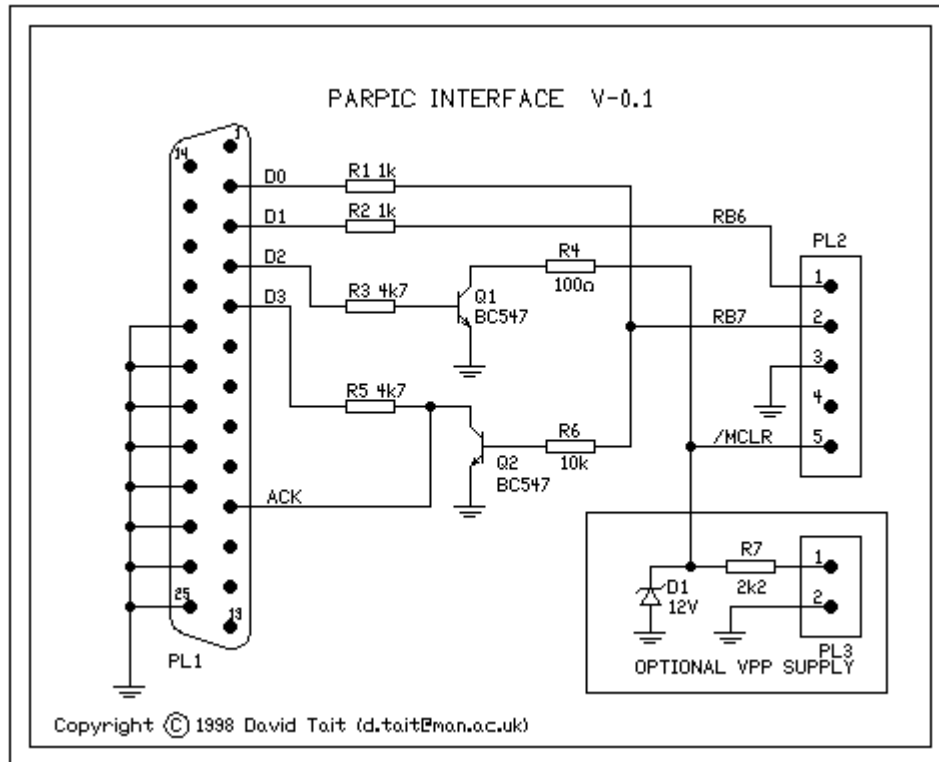


Figure 12 : Interface parallèle PARPIC INTERFACE de David Tait.

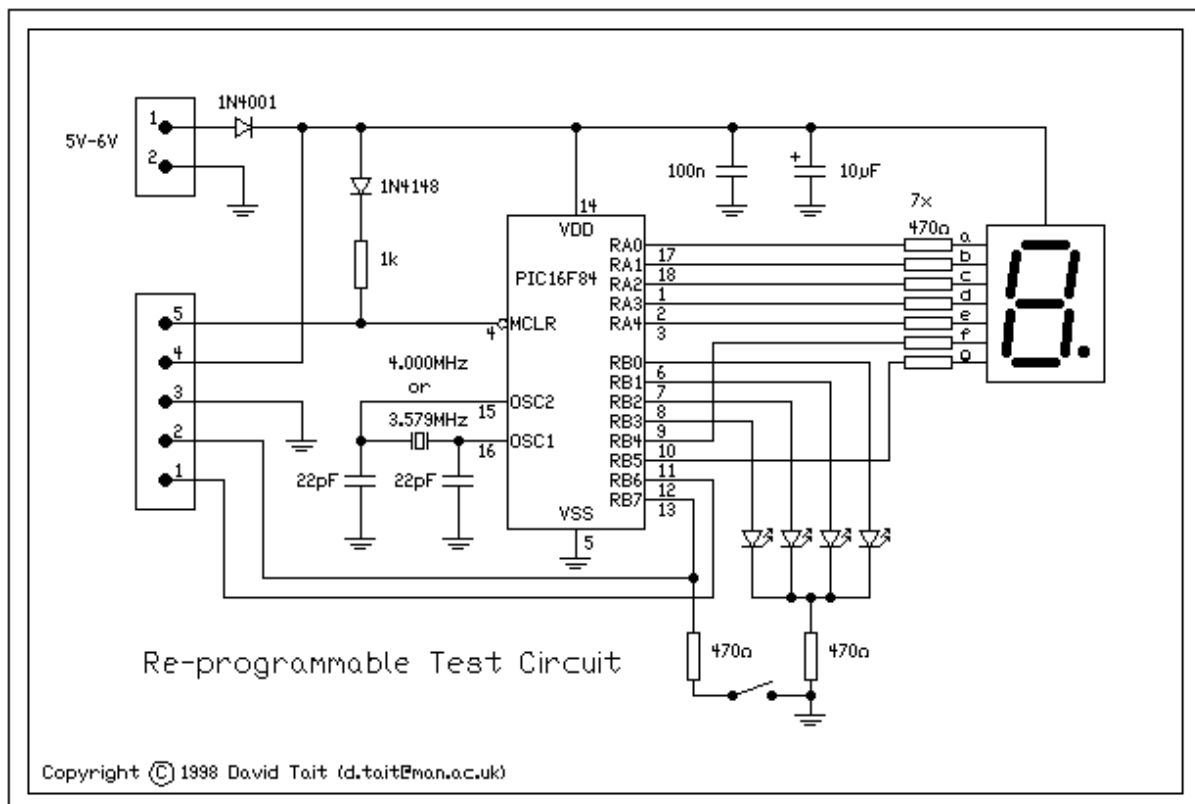


Figure 13 : Circuit de test REPROGRAMMABLE TEST CIRCUIT de David Tait.

Pour alimenter ce circuit nous devons utiliser une alimentation externe. Une alimentation stabilisée de laboratoire délivrant +/- 30v fait l'affaire.

Sachant que le microcontrôleur fonctionne sous une tension comprise entre +4v et +6v, un régulateur MC7805CT de la société MOTOROLA est utilisé pour fabriquer une tension de +5v. De plus, le microcontrôleur passe en mode programmation lorsqu'une tension comprise entre +12v et +14v est appliquée sur la broche MCLR/Vpp. Pour fabriquer cette tension, nous procédons de la même manière que précédemment en utilisant un régulateur MC7812CT, qui délivre +12v à sa sortie.

Ces deux circuits sont ensuite implanté aux deux circuits précédents, pour donner le schéma électrique de la figure 14, créer à l'aide du logiciel Isis de Labcenter Electronics.

✓ Une version d'évaluation du logiciel Isis est disponible sur le CD-ROM :

Répertoire : Isis

Référence : isis.exe

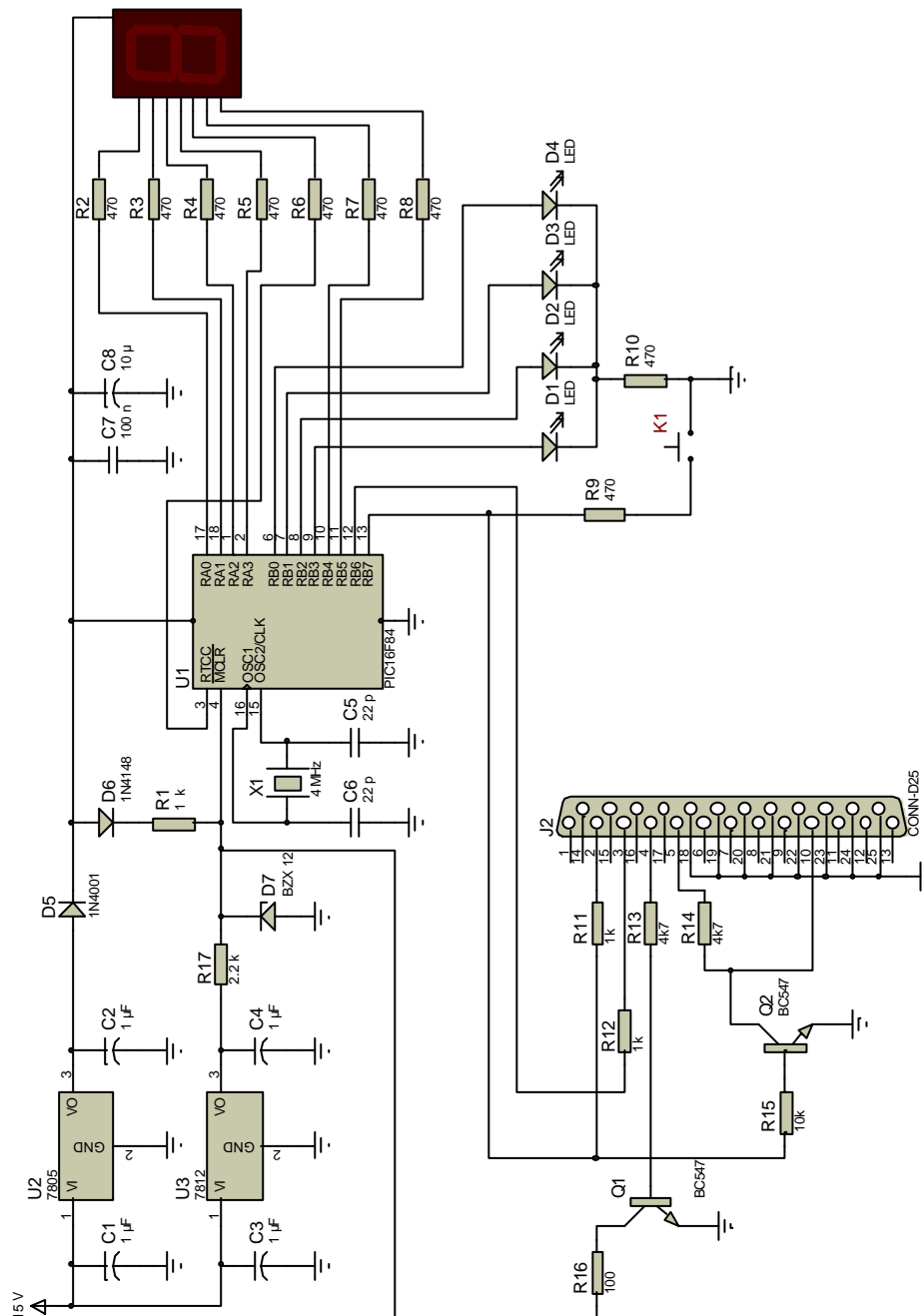


Figure 14 : Schéma électrique du programmeur PARPIC.

IV.1.2 – Réalisation du programmeur PARIPI

Après routage du circuit électrique précédent sous le logiciel Ares de la société Labcenter Electronics, nous obtenons les typons suivants (deux versions du même circuit) :

- ✓ Une version d'évaluation du logiciel Ares est disponible sur le CD-ROM :
Répertoire : Ares
Référence : ares.exe

Le typon de la figure 15 utilise un connecteur DB25 mâle, tandis que le typon de la figure 16 utilise un connecteur DB25 femelle.

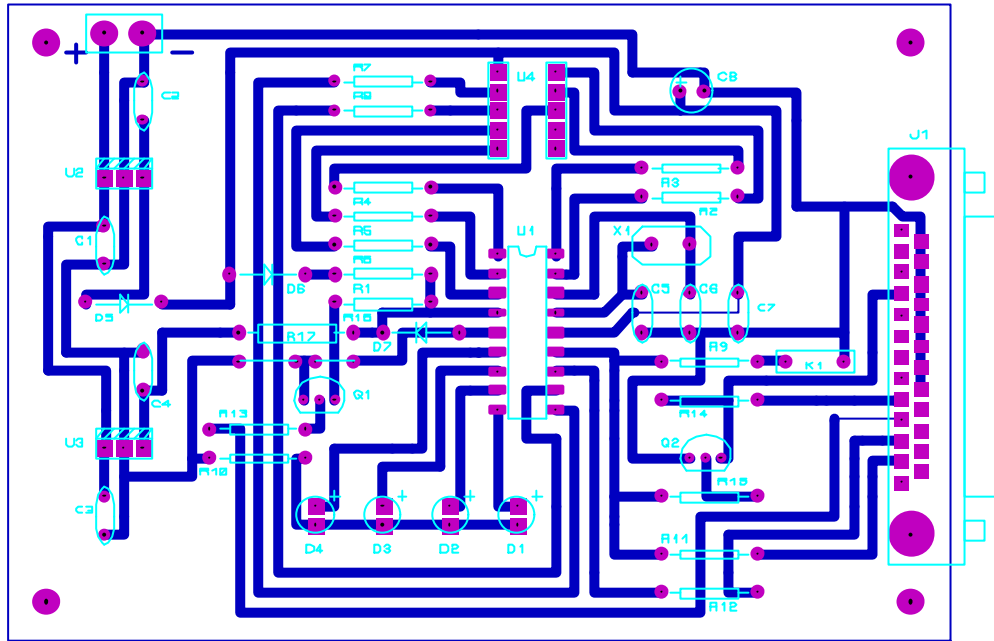


Figure 15 : Programmeur PARIPI DB25 mâle.

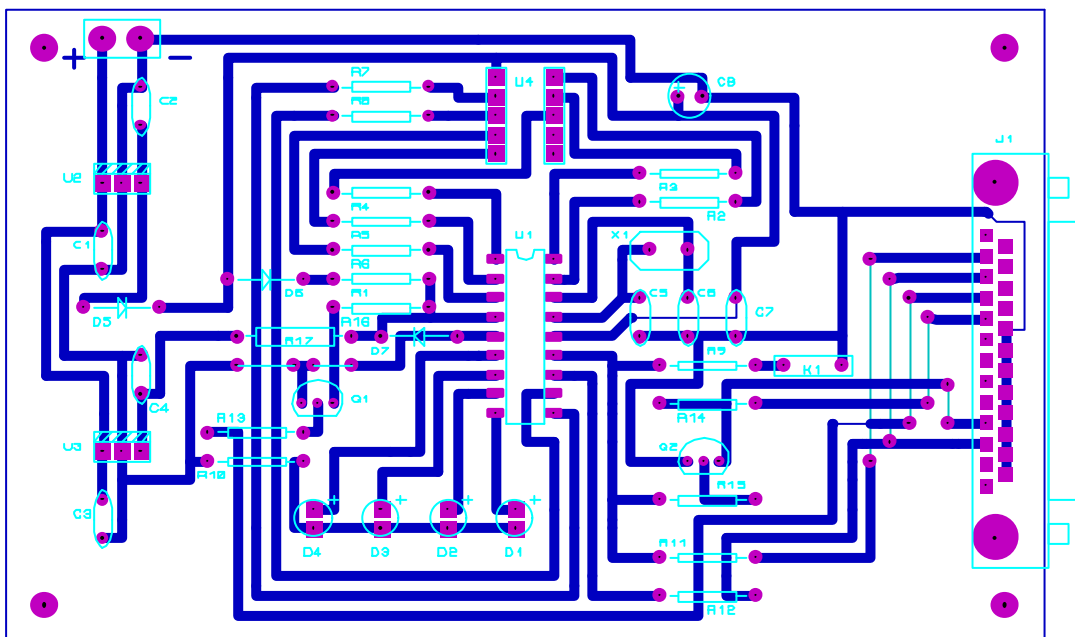


Figure 16 : Programmeur PARIPI DB25 femelle.

Le circuit à connecteur DB25 mâle permet son branchement directement sur le port parallèle du PC.

Le circuit qui au contraire utilise un connecteur DB25 femelle est raccordé au port parallèle du PC via un câble parallèle DB25 mâle droit.

✓ Les typons pour la réalisation sont disponibles en annexe A.

La figure 17 représente la nomenclature des composants utiles à la réalisation de l'un des programmeurs PARIPI (version DB25 femelle).

Quantité	Désignation	Caractéristiques	Prix TTC Unitaire	Prix TTC Total
4	C1, C2, C3, C4	Condensateur 1 µF électrochimique	1.90	7.60
2	C5, C6	Condensateur 22 pF céramique	0.53	1.06
1	C7	Condensateur 100 nF polyester	0.96	0.96
1	C8	Condensateur 10 µf électrochimique radial	1.40	1.40
3	R1, R11, R12	Résistance carbone 1KΩ ¼ W	0.15	0.45
9	R2 à R10	Résistance carbone 470 Ω ¼ W	0.15	1.20
2	R13, R14	Résistance carbone 4.7 KΩ ¼ W	0.15	0.30
1	R15	Résistance carbone 10 KΩ ¼ W	0.15	0.15
1	R16	Résistance carbone 100 Ω ¼ W	0.15	0.15
1	R17	Résistance carbone 2.2 KΩ ¼ W	0.15	0.15
4	D1, D2, D3, D4	DEL rouge Ø5mm	1.80	7.20
1	D5	Diode 1N4007	0.58	0.58
1	D6	Diode de commutation 1N4148	0.98	0.98
1	D7	Diode zener BZC12	0.59	0.59
1	X1	Quartz 4 MHz	16.20	16.20
2	Q1, Q2	Transistor de commutation BC547B	0.82	1.64
1	U1	Support DIL 18 broches	6.54	6.54
1	U2	Régulateur MC7805CT	8.38	8.38
1	U3	Régulateur MC7812CT	8.38	8.38
1	U4	Afficheur 7 segments anode commune	8.00	8.00
2	K1, K2	Interrupteur DIL pas de 7.6 mm	1.00	2.00
1	J1	Connecteur DB25 mâle ou femelle	18.74	18.74
1	B1	Bornier 2 bornes	2.47	2.47
1		Plaque Epoxy 130x100 mm	36.50	36.50
PRIX TOTAL TTC en Francs				131.62

Référence prix : Radiospares

Figure 17 : Nomenclature du programmeur PARIPI.

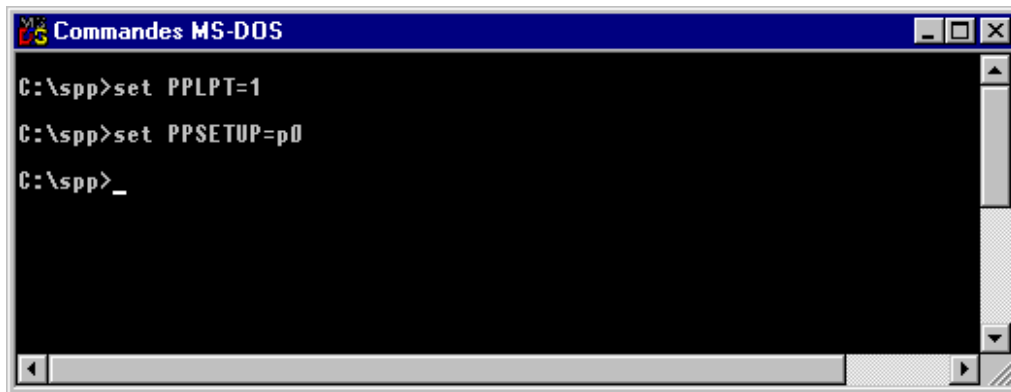
IV.1.3 – Fonctionnement du programmeur PARPIC

Pour faire fonctionner ce programmeur, nous utilisons un programme créé par David Tait fonctionnant sous environnement DOS ou commandes MS-DOS de Windows 95/98. Ce programme s'intitule **spp.exe**.

IV.1.3.1 – Initialisation de la transmission

Pour initialiser la transmission entre le programmeur **spp.exe** et l'ordinateur, il est nécessaire de créer deux variables PPSETUP et PPLPT.

- Sous commande MS-DOS taper **set PPLPT=1**.
- Puis **set PPSETUP=p0**.



```
Commandes MS-DOS
C:\spp>set PPLPT=1
C:\spp>set PPSETUP=p0
C:\spp>_
```

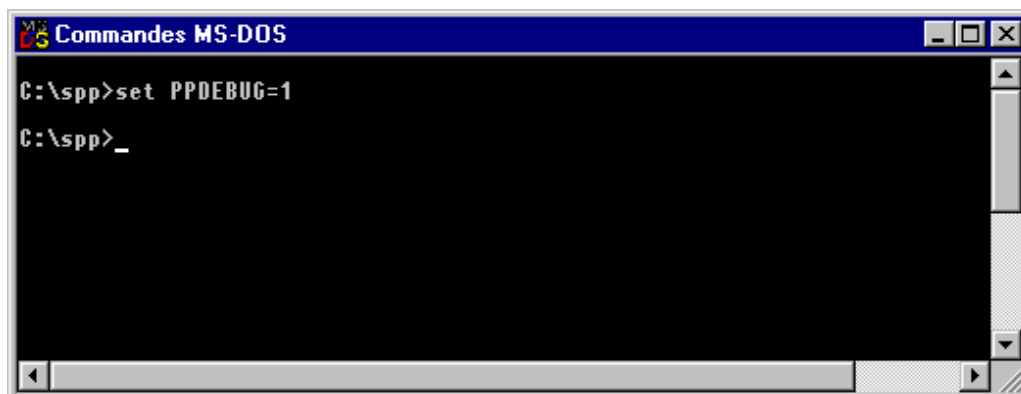
- Ces variables indiquent au programme **spp.exe** que nous utilisons le port parallèle n°1 de l'ordinateur et le programmeur PARPIC.

IV.1.3.1 – Mode DEBUG

Pour commencer, il est intéressant de connaître si oui ou non le programmeur réagit convenablement aux signaux qui lui seront envoyés par **spp.exe**.

Pour cela, il faut passer en mode DEBUG en initialisant la variable PPDEBUG à 1.

- Sous commande MS-DOS taper **set PPDEBUG=1**.



```
Commandes MS-DOS
C:\spp>set PPDEBUG=1
C:\spp>_
```

- Lancer le test en tapant **spp**.
- Suivre les instructions qui s'affichent en pressant la touche **Entrée**.

- Si tout fonctionne convenablement 2 messages (**input OK**) apparaissent.

```

Commandes MS-DOS
C:\spp>spp
Hardware setup: Parallel port (0378) with non-inverting buffers (delay
Debug mode entered ... (^C to exit)
Remove PIC ---
Run, RB6 low, RB7 low (input OK) ...
RB6 high ---
RB7 high (input OK) ...
Reset ---
Start over ... (^C to exit)

C:\spp>

```

- Si un des message indique (**input BAD**), le circuit n'est pas correct. Une vérification s'impose.

```

Commandes MS-DOS
C:\spp>spp
Hardware setup: Parallel port (0378) with non-inverting buffers (delay
Debug mode entered ... (^C to exit)
Remove PIC ---
Run, RB6 low, RB7 low (input OK) ...
RB6 high ---
RB7 high (input BAD) ...
Reset ---
Start over ... (^C to exit)

C:\spp>_

```

- Pour sortir du mode DEBUG, il faut réinitialiser PPDEBUG en le mettant à 0.
- Sous commandes MS-DOS tapez **set PPDEBUG=0**.

IV.1.3.2 – Mode programmation

Pour implanter un programme dans le microcontrôleur, suivre la démarche suivante :

- Taper **spp -n fichier.hex** Remplacer **fichier.hex** par le nom du fichier à implanter.

```

Commandes MS-DOS
C:\spp>spp -n digidie.hex

```

- Lorsque **spp.exe** le demande, alimenter le circuit et presser une touche.

```

Commandes MS-DOS - SPP
C:\spp>spp -n digidie.hex
PIC16F84 Programmer Version 0.1 Copyright (C) 1994-1998 David Tait.
Apply programming voltage ... hit any key to continue (^C to exit)

```

- Attendre la fin de la programmation.

```

Commandes MS-DOS
C:\spp>spp -n digidie.hex
PIC16F84 Programmer Version 0.1 Copyright (C) 1994-1998 David Tait.
Apply programming voltage ... hit any key to continue (^C to exit)
Programming ...
Setting config to ---X ...
Finished in 11 secs
C:\spp>

```

- Lorsque la programmation est terminée, éteindre l'alimentation puis débrancher le câble parallèle du programmeur.
- Si tout s'est déroulé normalement, le PIC doit fonctionner immédiatement lors de sa mise sous tension.

[IV.1.3.3 – Mode vérification](#)

Avec **spp.exe**, il est aussi possible de faire la comparaison entre le contenu du microcontrôleur et le fichier **.hex** qui lui a été implanté auparavant. Ceci permet de vérifier le bon déroulement de la programmation.

- Taper sous commandes MS-DOS : **spp -v fichier.hex**
- **fichier.hex** est le fichier implanté précédemment dans le microcontrôleur.
- Lorsque **spp.exe** le demande, alimenter le circuit et presser une touche.
- Si aucune erreur n'est détectée, aucun message particulier n'apparaît.



```
Commandes MS-DOS
C:\spp>spp -v digidie.hex
PIC16F84 Programmer Version 0.1 Copyright (C) 1994-1998 David Tait.
Apply programming voltage ... hit any key to continue (^C to exit)
Verifying ...
Finished in 1 sec
C:\spp>_
```

- Si une erreur est décelée, un message d'erreur l'indique par **Verify failed**.



```
Commandes MS-DOS
C:\spp>spp -v digidie.hex
PIC16F84 Programmer Version 0.1 Copyright (C) 1994-1998 David Tait.
Apply programming voltage ... hit any key to continue (^C to exit)
Verifying ...
spp: 0000: read 0183, wanted 0186
spp: Verify failed
C:\spp>_
```

- ✓ *Le programme spp.exe et sa documentation sont disponibles sur le CD-ROM :*
Répertoire : Spp

IV.2 – Etude et réalisation du programmeur QUICK AND DIRTY

IV.2.1 – Etude du programmeur QUICK AND DIRTY de David Tait

Pour fonctionner, ce circuit utilise aussi un programme créé par David Tait fonctionnant sous DOS ou commandes MS-DOS de Windows 95/98.

Le schéma du circuit est représenté ci-dessous en figure 18 :

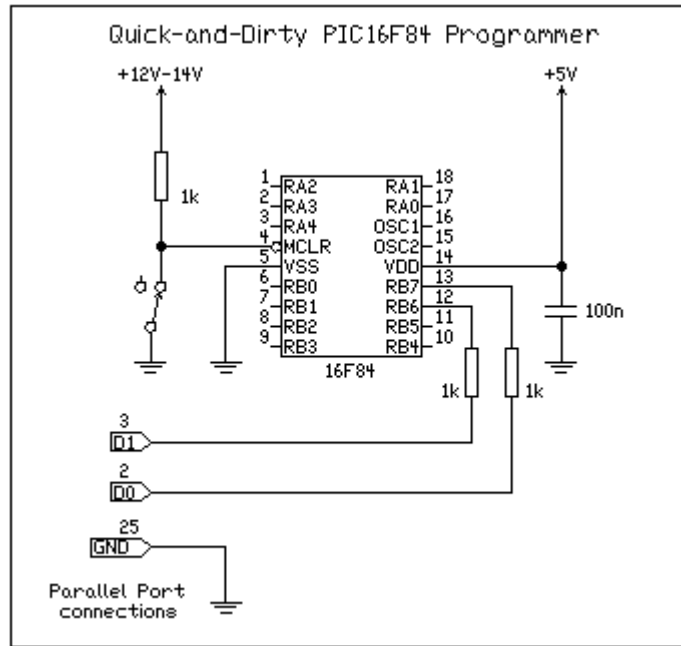


Figure 18 : Programmeur QUICK-AND-DIRTY de David Tait.

Comme pour le circuit PARNIC, le schéma précédent a été modifié par l'ajout de deux régulateurs et leurs condensateurs de découplage, afin de fabriquer les +5v et +12v nécessaires au fonctionnement et à la programmation du microcontrôleur.

La figure 19 suivante représente le nouveau schéma électrique du programmeur.

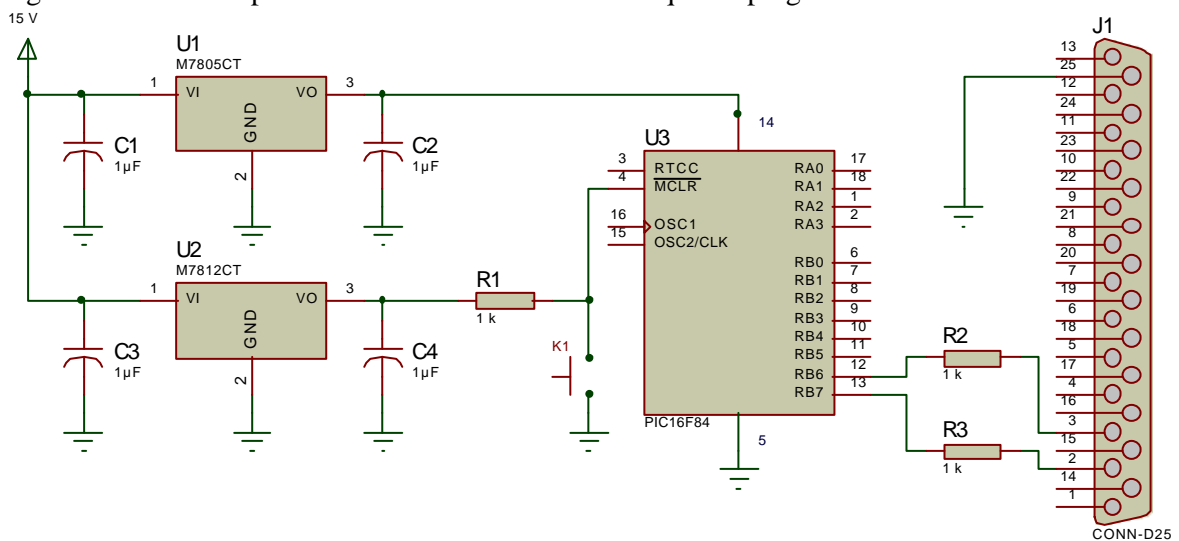


Figure 19 : Schéma électrique du programmeur QUICK AND DIRTY.

IV.2.2 – Réalisation du programmeur QUICK AND DIRTY

Après routage du circuit électrique précédent sous le logiciel Ares, nous obtenons les typons suivants : (figure 20 et 21)

Comme pour le montage PARPIC, deux versions de ce programmeur sont possibles. L'une dispose d'une prise DB25 mâle et l'autre d'une prise DB25 femelle.

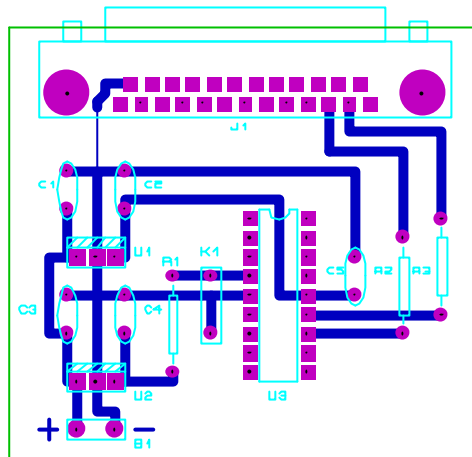


Figure 20 : Programmeur QUICK AND DIRTY DB25 mâle.

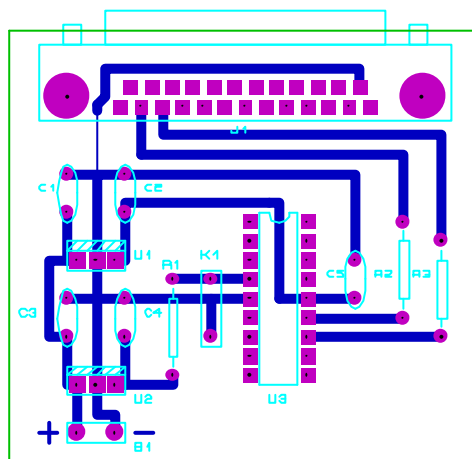


Figure 21 : Programmeur QUICK AND DIRTY DB25 femelle.

✓ Les typons pour la réalisation des circuits sont disponibles en annexe B du rapport.

La figure 22 représente la nomenclature des composants utiles à la réalisation de l'un des programmeurs QUICK AND DIRTY.

Quantité	Désignation	Caractéristiques	Prix TTC Unitaire	Prix TTC Total
4	C1, C2, C3, C4	Condensateur 1µF électrochimique	1.90	7.60
1	C5	Condensateur 100 nF polyester	0.96	0.96
3	R1, R2, R3	Résistance carbone 1KΩ ¼ W	0.15	0.45
1	U1	Régulateur MC7805CT	8.38	8.38
1	U2	Régulateur MC7812CT	8.38	8.38
1	U3	Support DIL 18 broches	6.54	6.54
1	K1	Interrupteur DIL pas de 7.6 mm	1.00	1.00
1	J1	Connecteur DB25 femelle	18.74	18.74
1	B1	Bornier 2 bornes	2.47	2.47
1		Plaque Epoxy 1 face 160x100 mm	36.50	36.50
PRIX TOTAL TTC en Francs				91.02

Référence prix : Radiospares

Figure 22 : Nomenclature du programmeur QUICK AND DIRTY.

IV.2.3 – Fonctionnement du programmeur QUICK AND DIRTY

Pour faire fonctionner ce programmeur, nous devons procéder de la même manière que pour le programmeur PARPIC en utilisant un programme fonctionnant sous DOS. A la différence de **spp.exe**, ce programme permet seulement la programmation. De ce fait, on ne peut l'utiliser que pour lire le contenu d'un microcontrôleur.

Ce programme s'intitule **pp.exe**.

IV.2.3.1 – Initialisation de la transmission

Pour initialiser la transmission entre le programmeur, **pp.exe** et l'ordinateur, il est nécessaire de créer deux variables PPSETUP et PPLPT.

- Sous commande MS-DOS taper **set PPLPT=1** puis **set PPSETUP=3**.

```

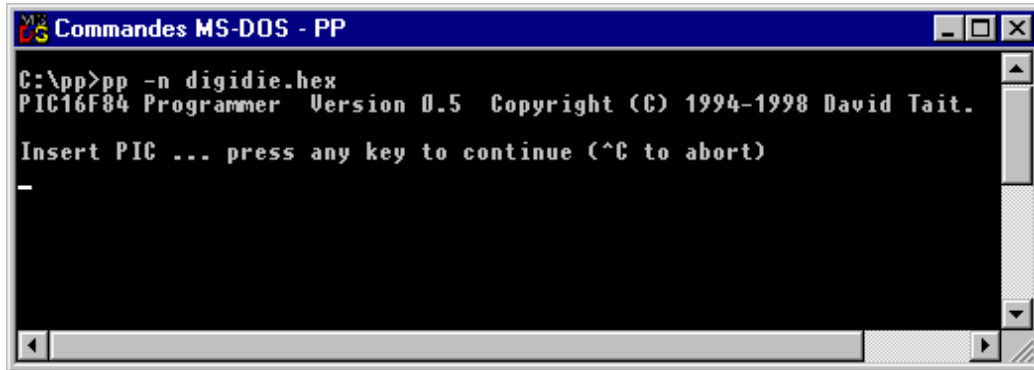
Commandes MS-DOS
C:\pp>set PPLPT=1
C:\pp>set PPSETUP=3
C:\pp>
  
```

- Ces variables indiquent au programme **pp.exe** que nous utilisons le port parallèle n°1 de l'ordinateur et le programmeur QUICK AND DIRTY.

IV.2.3.2 – Mode programmation

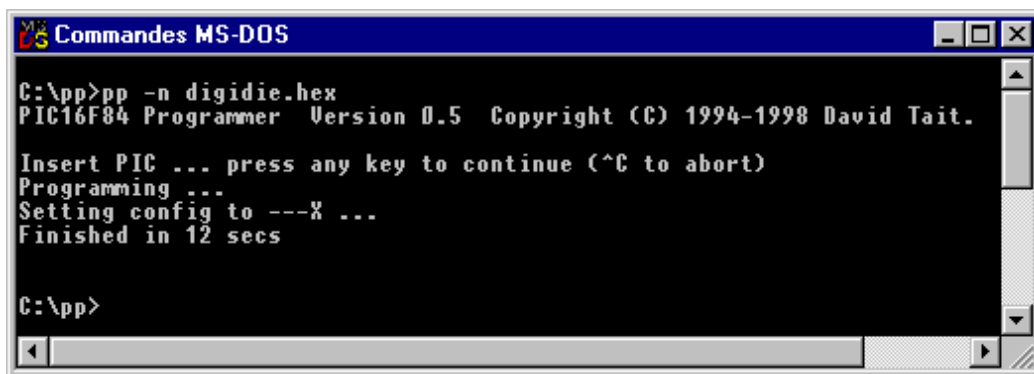
Pour implanter un programme dans le microcontrôleur, suivre la démarche suivante :

- Taper **pp -n fichier.hex** Remplacer **fichier.hex** par le nom du fichier à implanter.



```
Commandes MS-DOS - PP
C:\app>pp -n digidie.hex
PIC16F84 Programmer Version 0.5 Copyright (C) 1994-1998 David Tait.
Insert PIC ... press any key to continue (^C to abort)
_
```

- Quand **pp.exe** indique **Insert PIC ... press any key to continue**, ouvrir l'interrupteur K1 pour mettre MCLR/Vpp à +12v et presser une touche quelconque.
- Attendre la fin de la programmation.



```
Commandes MS-DOS
C:\app>pp -n digidie.hex
PIC16F84 Programmer Version 0.5 Copyright (C) 1994-1998 David Tait.
Insert PIC ... press any key to continue (^C to abort)
Programming ...
Setting config to ---X ...
Finished in 12 secs

C:\app>
```

- Lorsque la programmation est terminée, fermer l'interrupteur K1 (MCLR/Vpp à la masse).
- Eteindre l'alimentation.

Malgré son fonctionnement très simple, un troisième programmeur a été réalisé.

Comme le programmeur PARPIC, il peut lire, écrire et effacer la mémoire des microcontrôleurs. Son intérêt réside dans sa manipulation : il se pilote à la souris sous environnement Windows.

- ✓ *Le programme pp.exe et sa documentation sont disponibles sur le CD-ROM :*
Répertoire : Pp

IV.3 – Etude et réalisation du programmeur PICPROG 2000

IV.3.1 – Etude du programmeur PICPROG 2000 de Jacques Weiss

Ce programmeur se distingue des ses prédécesseurs puisqu'il ne nécessite pas d'alimentation spécifique (+5v/0v suffit). De plus, son logiciel d'exploitation est très convivial et simple à utiliser. La figure suivante représente le schéma électrique du programmeur.

Figure 23 : Schéma électrique du programmeur PICPROG 2000.

IV.3.3 – Fonctionnement du programmeur PICPROG 2000

Pour faire fonctionner ce programmeur, nous devons utiliser le logiciel créé par Jacques Weiss. Ce logiciel, Universal PIC Programmer, fonctionne sous Windows 95/98. Il permet d'écrire, lire, vérifier et effacer le contenu des microcontrôleurs. Son utilisation est aisée, puisque toutes les commandes sont représentées sur l'interface graphique (figure 26).

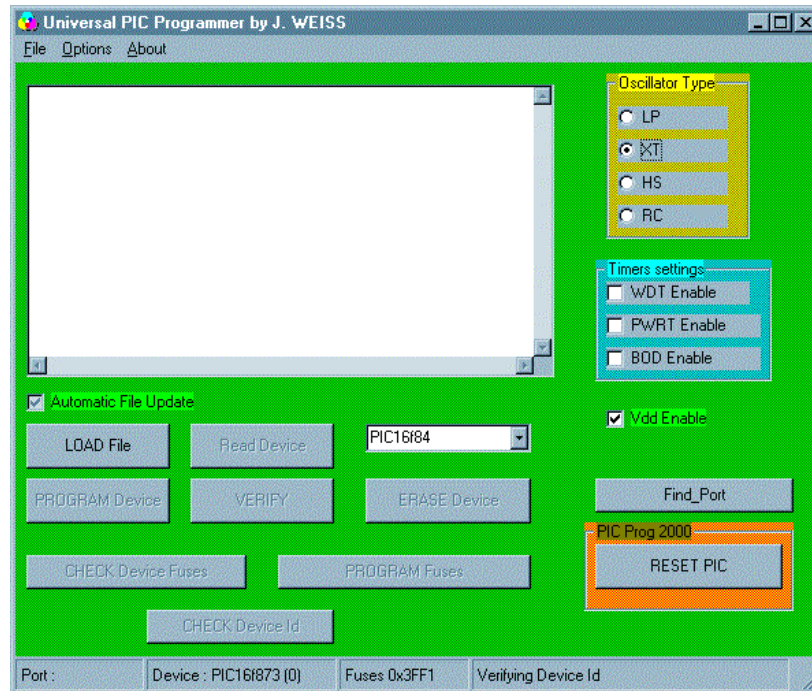
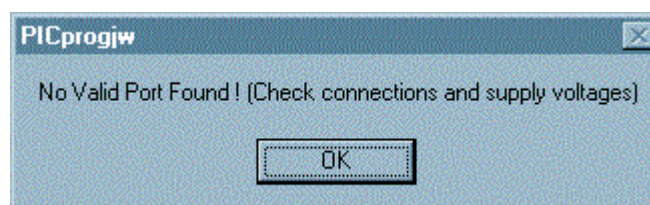
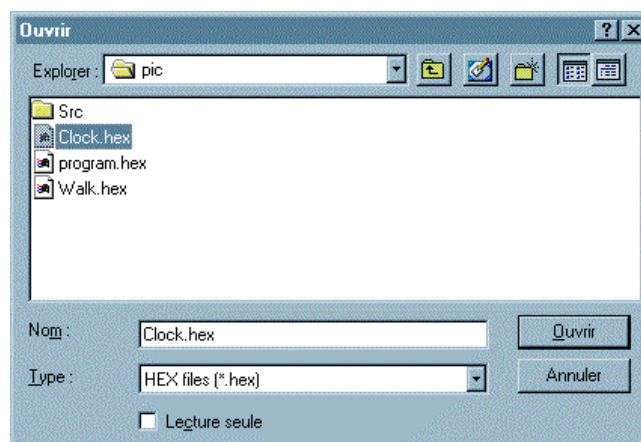


Figure 26 : Interface graphique du logiciel Universal PIC Programmer.

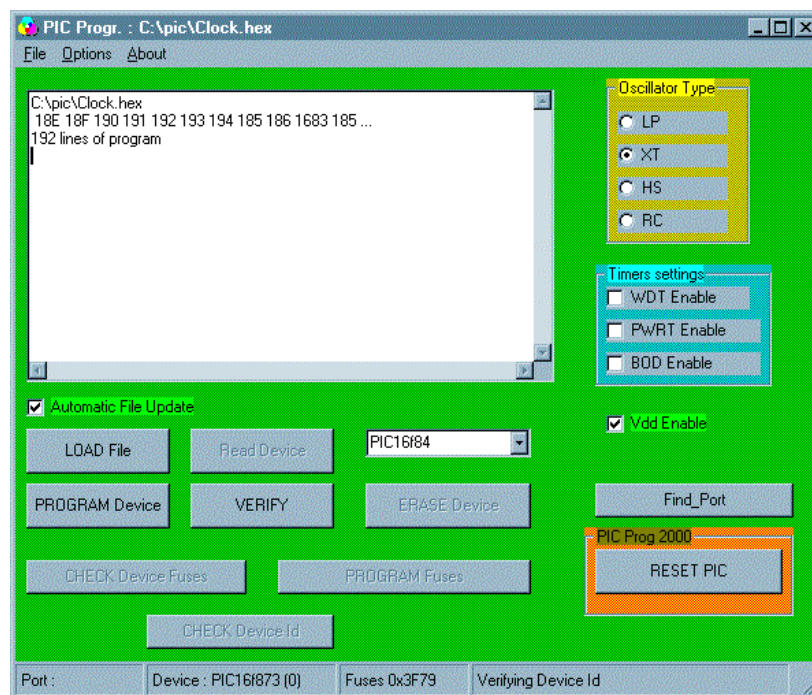
- Il est recommandé d'alimenter le programmeur avant de lancer le logiciel sous peine de voir apparaître le message d'erreur suivant.



- Pour charger un programme dans le microcontrôleur, cliquer sur **LOAD**. La fenêtre suivante apparaît :



- A partir de cette fenêtre, choisir le fichier **.hex** à implanter et cliquer sur **OUVRIR**.
- Le fichier **.hex** est alors chargé en mémoire et prêt à être transféré dans le microcontrôleur.



- Cliquer sur **PROGRAM Device** pour charger le fichier **.hex** dans le microcontrôleur.
- Pour vérifier le bon déroulement de la programmation, cliquer sur **VERIFY**.

Pour les autres manipulations, procéder de la même manière en utilisant les boutons de l'interface graphique.

- ✓ *Le logiciel Universal PIC Programmer est disponible sur le CD-ROM :*
Répertoire : Picprog/soft
Référence : setup.exe

V – Application : codage de voix ADPCM

Il y a quelques temps, pour ajouter le contrôle vocal à un système intégré, il était nécessaire d'utiliser un processeur DSP. Aujourd'hui un microcontrôleur 8 bits et des algorithmes de compression de la voix suffisent.

V.1 – Origine de cette application

Un article intitulé "Implementing speech on 8bit microcontroller", écrit par Monsieur Roger Richey et paru dans le magazine Embedded Systems d'avril 2000, a été le point de départ de ce projet.

Monsieur Rodger Richey, ingénieur américain travaillant pour la société Microchip, explique la possibilité d'implanter la parole dans les microcontrôleurs 8 bits en utilisant le codage ADPCM.

✓ *Une copie de cet article est présentée en annexe D.*

Un second document (note d'application AN643), disponible sur le site Internet de la société Microchip, traitant de la restitution de voix codée en ADPCM, a été une aide précieuse dans la réalisation de ce projet.

✓ *Le document AN643 est disponible sur le CD-ROM :*
Répertoire : AN643
Référence : AN643.pdf

V.2 – Cahier des charges

Le cahier des charges de cette application se résume à ces quelques lignes.

- Utiliser un microcontrôleur PIC.
- Utiliser l'algorithme ADPCM.
- Enregistrer et coder de la voix à partir d'un microphone.
- Décoder et restituer la voix sur haut-parleur.

V.3 – Etude théorique de l'application

V.3.1 – Première approche

En s'appuyant sur la documentation de Rodger Richey et sur certains documents techniques, il en ressort quelques observations :

- L'encodage et le décodage ADPCM nécessitent une certaine vitesse de calcul, l'horloge du microcontrôleur doit donc être assez importante.
- Pour échantillonner le signal d'entrée et le traiter, deux solutions existent. La première consiste à utiliser un convertisseur analogique numérique relié au microcontrôleur. La seconde, à utiliser directement le convertisseur analogique numérique de certain microcontrôleur.

- Le gain d'un microphone n'étant pas très élevé, un préamplificateur doit donc être utilisé pour l'amplifier.
- L'échantillonnage s'effectue à la fréquence de 8 KHz, le signal doit donc être filtré par un filtre passe bas de fréquence de coupure 4 KHz (théorème de Shannon).
- Pour pouvoir enregistrer et lire indéfiniment, la mémoire de stockage des codes ADPCM doit être de type RAM.
- Pour restituer le signal codé en un signal audible, deux solutions sont possibles. La première consiste à utiliser un convertisseur numérique analogique en sortie du microcontrôleur. Et la seconde à faire appel au module PWM (Pulse Width Modulation : Modulation à largeur d'impulsion) de certain microcontrôleur.
- Le signal de sortie doit être audible et restitué sur haut-parleur, un amplificateur audio doit donc être aussi incorporé au circuit.

V.3.2 - Schéma de principe de la première approche

Le schéma de principe, donné en figure 27, ci-dessous, résume la première approche.

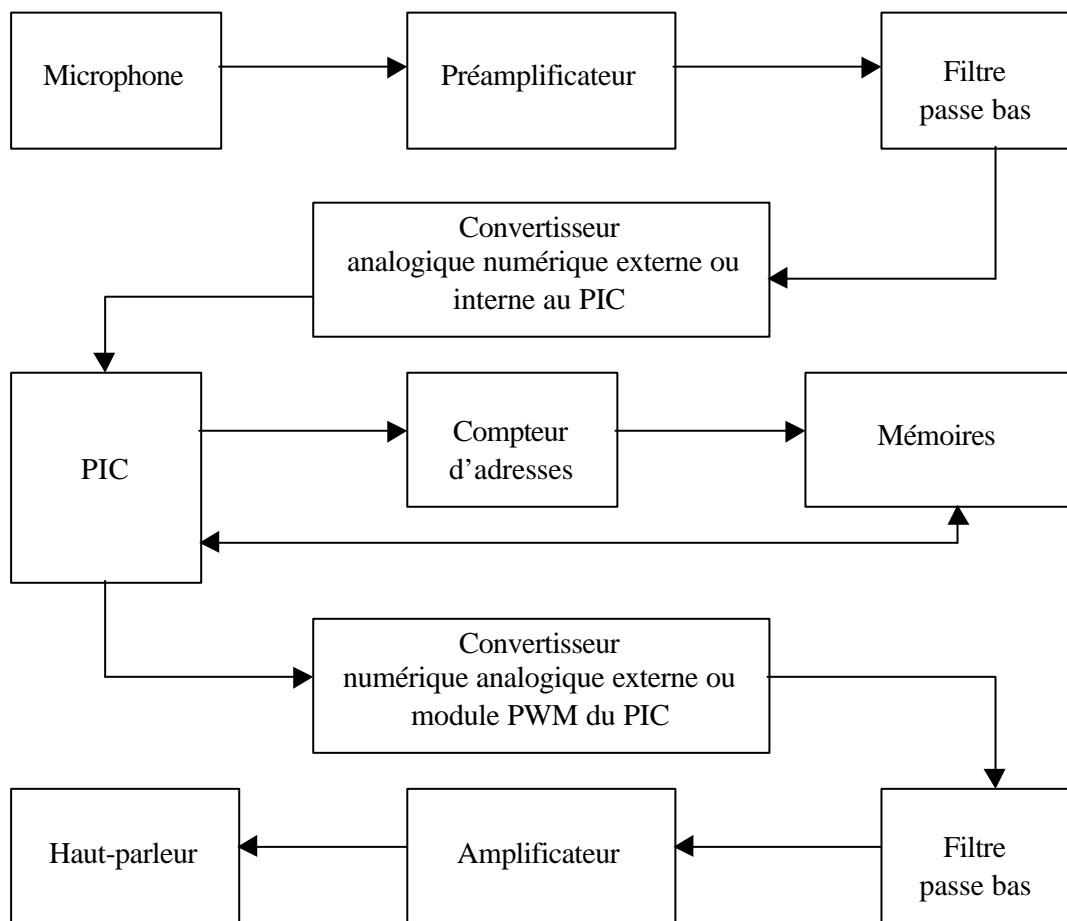


Figure 27 : Schéma de principe de la première approche.

V.4 – Etude matérielle

Suite à étude des différentes solutions proposées dans la première approche, des choix s'imposent.

Le choix du microcontrôleur s'est arrêté sur le PIC 16C72 de Microchip, dont les caractéristiques sont les suivantes :

Microcontrôleur PIC 16C72 :

- Type : PIC 16C72 Microchip
- Périphériques : Convertisseur analogique/numérique 8 bits
Module PWM 10 bits
3 Modules timers
Communication série SPI/I²C
- Horloge : 20 MHz
- Mémoire RAM : 128 x 8 bits
- Mémoire ROM : 2k x 14 bits
- Nombre d'entrées/sorties : 22

Pour la numérisation du signal d'entrée, le convertisseur analogique/numérique du microcontrôleur est retenu.

Pour la conversion numérique/analogique, le choix s'est porté sur le module PWM du microcontrôleur.

La mémoire de stockage choisie est une SRAM TC551001CP de 128 Ko de marque Toshiba.

Le choix du préamplificateur s'est porté sur le SSM2017 de Analog Devices dont le gain maximal peut atteindre 70 dB.

Les filtres sont des filtres passe bas de type Butterworth du 4^{ème} ordre, de fréquence de coupure 4KHz.

Un compteur binaire 12 bits CD4040B de Texas Instruments suivi d'un compteur binaire 7 bits CD4024B sont choisis pour remplir la fonction de compteur d'adresses.

Un amplificateur audio LM386 de National Semiconductor est choisi pour attaquer le haut-parleur.

V.4.1 – Etude du préamplificateur

Cet ensemble est réalisé à partir du préamplificateur SSM2017 de Analog Devices. Son principal intérêt réside dans sa performance et son gain ajustable.

Schéma électrique :

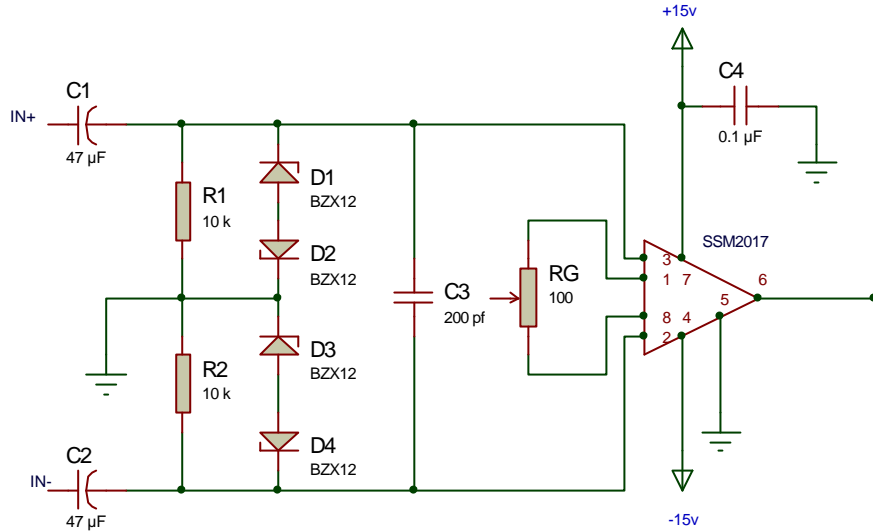


Figure 28 : Schéma électrique du préamplificateur.

Le potentiomètre placé entre les bornes 1 et 8 du SSM2017 permet de régler le gain du préamplificateur.

Le gain G suit la fonction suivante :

$$G = \frac{10k\Omega}{RG} + 1$$

et

$$RG = \frac{10k\Omega}{G - 1}$$

Par exemple, pour obtenir un gain de 50 dB, RG doit être égal à 32 Ω :

$$50dB = 20 \log G$$

d'où

$$G = 10^{\frac{50}{20}} = 316$$

donc

$$RG = \frac{10k\Omega}{316 - 1} = 31.74 \approx 32\Omega$$

- ✓ Les résultats des tests du préamplificateur se trouvent en annexe E.
- ✓ La documentation technique du SSM2017 est disponible sur le CD-ROM :
Répertoire : Doc
Référence : SSM2017.pdf

V.4.2 – Etude du filtre passe bas d'entrée

Nous souhaitons échantillonner le signal d'entrée à la fréquence F_e de 8 KHz. D'après le théorème de Shannon, la fréquence d'échantillonnage F_e doit être au moins le double de la fréquence maximale du signal à échantillonner F_{max} :

$$F_e \geq 2 F_{max}$$

La fréquence maximale admissible F_{max} à l'entrée du CAN doit donc être égale à 4 KHz.

Pour ne laisser passer que les fréquences inférieures à 4KHz, la fréquence de coupure F_c du filtre passe bas doit être, elle aussi, égale à 4KHz.

L'ordre du filtre et son type sont aussi importants. C'est pour cette raison que le filtre est du 4^{ème} ordre et de type Butterworth.

Schéma électrique :

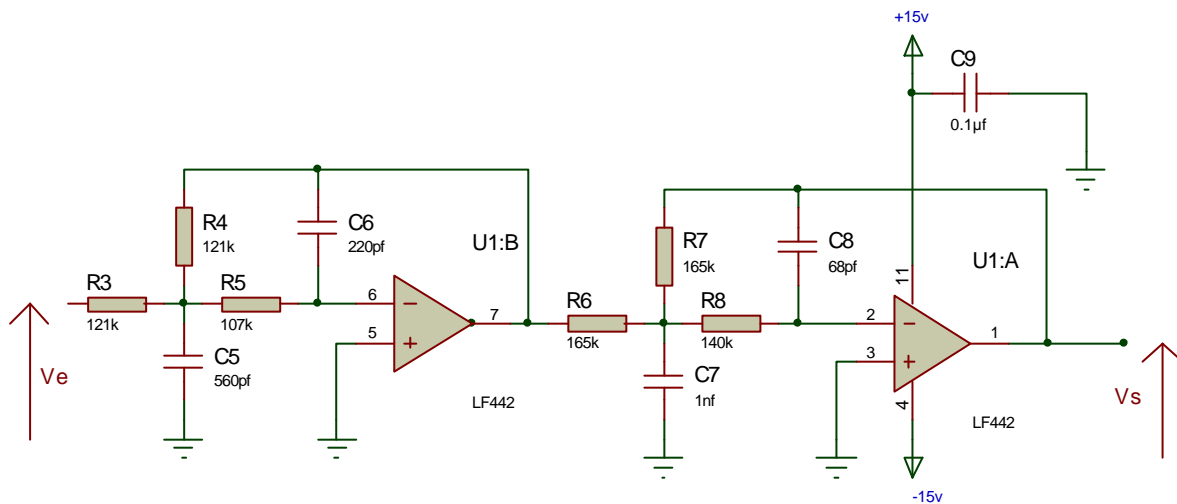


Figure 29 : Schéma électrique du filtre passe bas.

Le filtre est réalisé à partir de deux filtres passe bas du second ordre (structure de RAUCH) montés en cascade.

- ✓ La courbe de gain théorique et pratique du filtre ($20 \log \frac{V_s}{V_e}$) se trouvent en annexe F.
- ✓ La documentation technique du LF442 est disponible sur le CD-ROM :
Répertoire : Doc
Référence : LF442.pdf

V.4.3 – Etude du Convertisseur Analogique Numérique (CAN)

Nous utilisons, ici, le module Convertisseur Analogique Numérique du microcontrôleur PIC 16C72. Celui-ci peut convertir sur 8 bits, les valeurs de 5 signaux analogiques présents sur chacune des 5 entrées du CAN du PIC.

Sachant que nous devons échantillonner à la fréquence de 8KHz, le CAN doit être configuré de façon à ce qu'il exécute chaque conversion à cette fréquence, tout en respectant les temps d'acquisition et de conversion, nécessaires à l'échantillonnage.

La référence de tension utilisée pour la conversion peut être définie comme étant la tension d'alimentation du PIC ou la tension appliquée sur la broche RA3/AN3/Vref du microcontrôleur.

Schéma électrique de l'ensemble CAN/tension de référence :

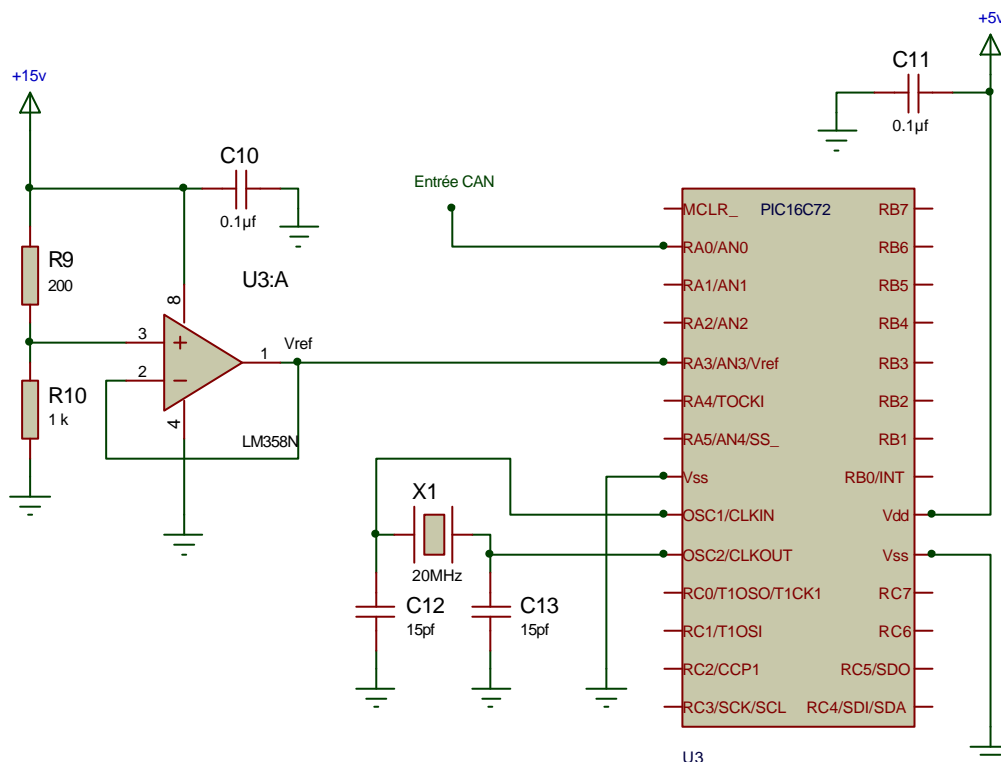


Figure 30 : Schéma électrique de la tension de référence du CAN.

Dans l'application, la tension Vref est réalisée à partir d'un pont de résistance suivi d'un suiveur de tension. Le suiveur, réalisé à partir d'un LM358N de National Semiconductor, permet d'attaquer l'entrée du CAN avec une faible impédance.

Seule la broche RA0 du PORTA est choisie comme entrée du CAN.

- ✓ La documentation technique du PIC16C72 est disponible sur le CD-ROM :
Répertoire : Doc
Référence : PIC16C72.pdf

V.4.4 – Etude du compteur d'adresses

Comme nous l'avons vu dans la première approche, la mémoire choisie pour enregistrer les codes ADPCM peut contenir jusqu'à 128 Ko de données. Ce qui représente une taille assez importante pour un microcontrôleur, sachant qu'il doit pouvoir l'utiliser dans sa totalité.

Quelles ressources sont nécessaires pour adresser la totalité de cette mémoire ?

$$128 \text{ Ko} = 128 \times 1024 = 131072 \text{ octets}$$

Pour adresser autant d'octets, il faut 17 bits puisque $2^{17} = 131072$. En utilisant simplement les ports du microcontrôleur, il en faudrait au moins trois, ce qui est impossible ou tout simplement impensable.

D'où l'idée de concevoir un compteur d'adresses constitué de compteurs binaires montés en cascade, comme le montre le schéma, ci dessous.

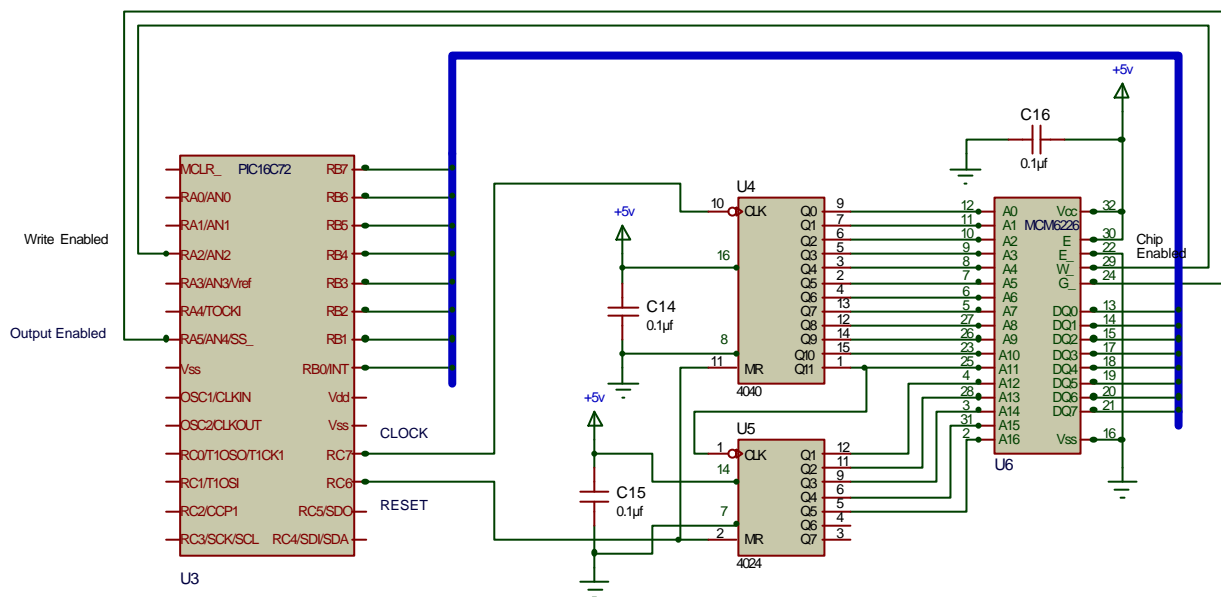


Figure 31 : Schéma électrique du compteur d'adresses.

Ce compteur fait appel à un compteur binaire CD4040B et CD4024B, respectivement de 12 et 7 bits de Texas Instruments.

L'intérêt de ce système réside dans sa simplicité et le peu de ressources nécessaires à son fonctionnement. En effet, deux lignes du PORTC du microcontrôleur suffisent pour le contrôler. Le front descendant de chaque impulsion envoyée sur RC7 incrémente le compteur, ce qui sélectionne l'adresse de l'octet suivant. Une impulsion envoyée sur RC6 provoque le RESET du compteur qui revient à zéro.

Comme nous pouvons le voir, seules les 17 premières sorties de ce compteur sont connectées à la RAM. Le RESET du compteur est provoqué logiquement après que les 131072 adresses aient été comptées.

- ✓ La documentation technique du CD4040B et CD4024B est disponible sur le CD-ROM :
Répertoire : Doc
Référence : CD40XXB.pdf

V.4.5 – Etude du module PWM

Pour restituer le signal codé en ADPCM en signal analogique, il faut effectuer une conversion numérique/analogique. Comme nous l'avons vu au début du chapitre V, nous avons choisi d'utiliser le module PWM du microcontrôleur pour effectuer cette opération.

Qu'est ce qu'un module PWM ?

Un module PWM, (Pulse Width Modulation) signifiant Modulation à largeur d'impulsion, permet de fabriquer un signal numérique modulé en largeur d'impulsion.

Ce module est souvent utilisé dans les applications utilisant la parole puisqu'il peut se substituer au convertisseur numérique/analogique plus coûteux, lorsqu'il est associé à un filtre passe bas.

Comment obtenir des signaux analogiques à partir de signaux PWM ?

La conversion de signaux PWM en signaux analogiques nécessite l'utilisation d'un filtre passe bas dont les caractéristiques dépendent de la fréquence du signal PWM.

Qu'est ce qu'un signal PWM ?

Dans un signal PWM typique, comme le montre la figure 32, la fréquence du signal est toujours constante alors que le rapport cyclique varie (de 0% à 100%) en fonction de l'amplitude du signal d'origine.

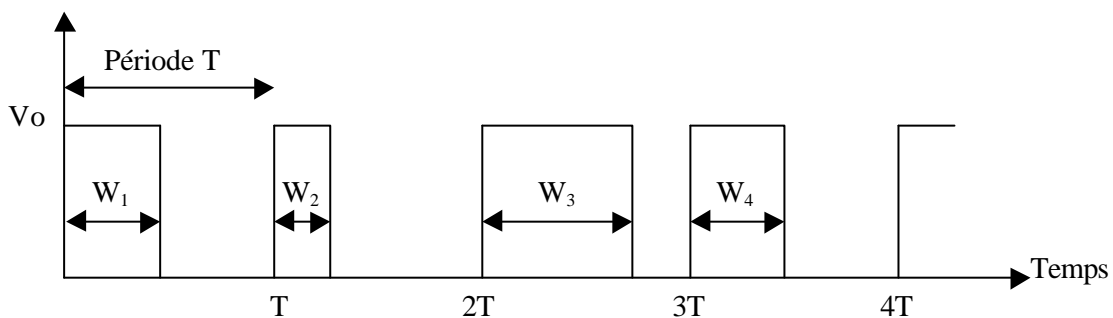


Figure 32 : Signal PWM.

Le spectre de Fourier d'un tel signal, montre qu'il existe un certain nombre de pics d'amplitude variable aux fréquences $F = \frac{K}{T}$, avec K un réel impair.

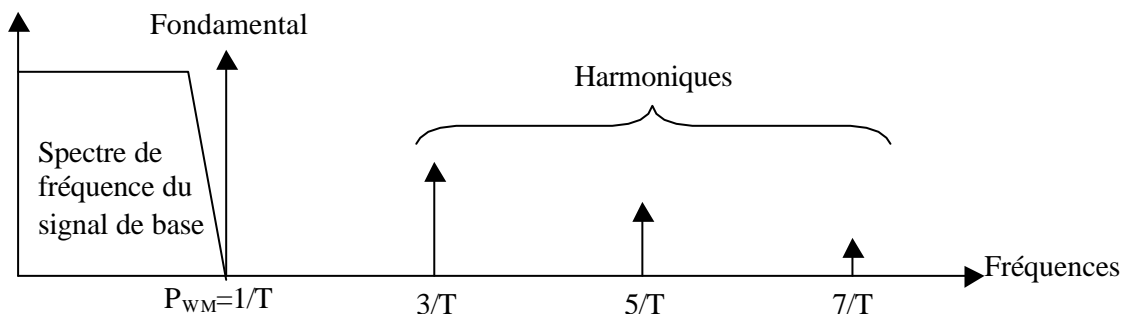


Figure 33 : Spectre de fréquence d'un signal PWM.

Le premier pic de fréquence $F_{PWM} = \frac{1}{T}$ représente le fondamental du signal PWM. Les autres pics de fréquences, $F = \frac{3}{T}$, $F = \frac{5}{T}$, ..., représentent les harmoniques du signal. Ces harmoniques considérés comme étant du bruit doivent être éliminés. Ceci implique le filtrage du signal PWM.

Dans l'application, nous souhaitons fabriquer un signal analogique restituant la parole à partir d'un signal PWM. Sachant que le signal précédemment numérisé possède une bande passante de 4KHz (après filtrage), la bande passante du signal désiré F_{BW} en sortie du module PWM doit être identique.

Donc :

$$F_{BW} \leq (F_{PWM} = \frac{1}{T})$$

Si F_{BW} est égal à F_{PWM} , le filtre passe bas doit être énormément sélectif, ce qui est difficile et assez coûteux à réaliser. Aussi, pour une réalisation plus simple, le filtre doit se rapprocher de la figure 32.

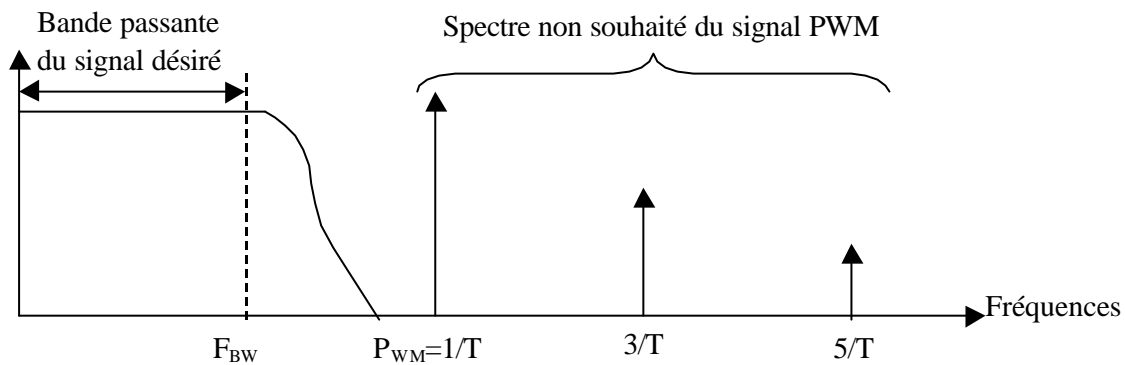


Figure 34 : Caractéristique du filtre passe bas.

Ce qui impose :

$$F_{BW} \ll F_{PWM}$$

ou

$$F_{PWM} \gg F_{BW}$$

soit

$$F_{PWM} = K \times F_{BW}$$

où K est une constante telle que $K \gg 1$.

De plus, plus la fréquence F_{PWM} est élevée, meilleure est la réjection du bruit.

D'après toutes ces données, nous choisissons pour F_{PWM} la valeur de 32 KHz.

Quant au filtre, il reprend les mêmes caractéristiques que le filtre passe bas vu précédemment : filtre passe bas du 4^{ème} ordre de type Butterworth.

Schéma électrique de l'ensemble :

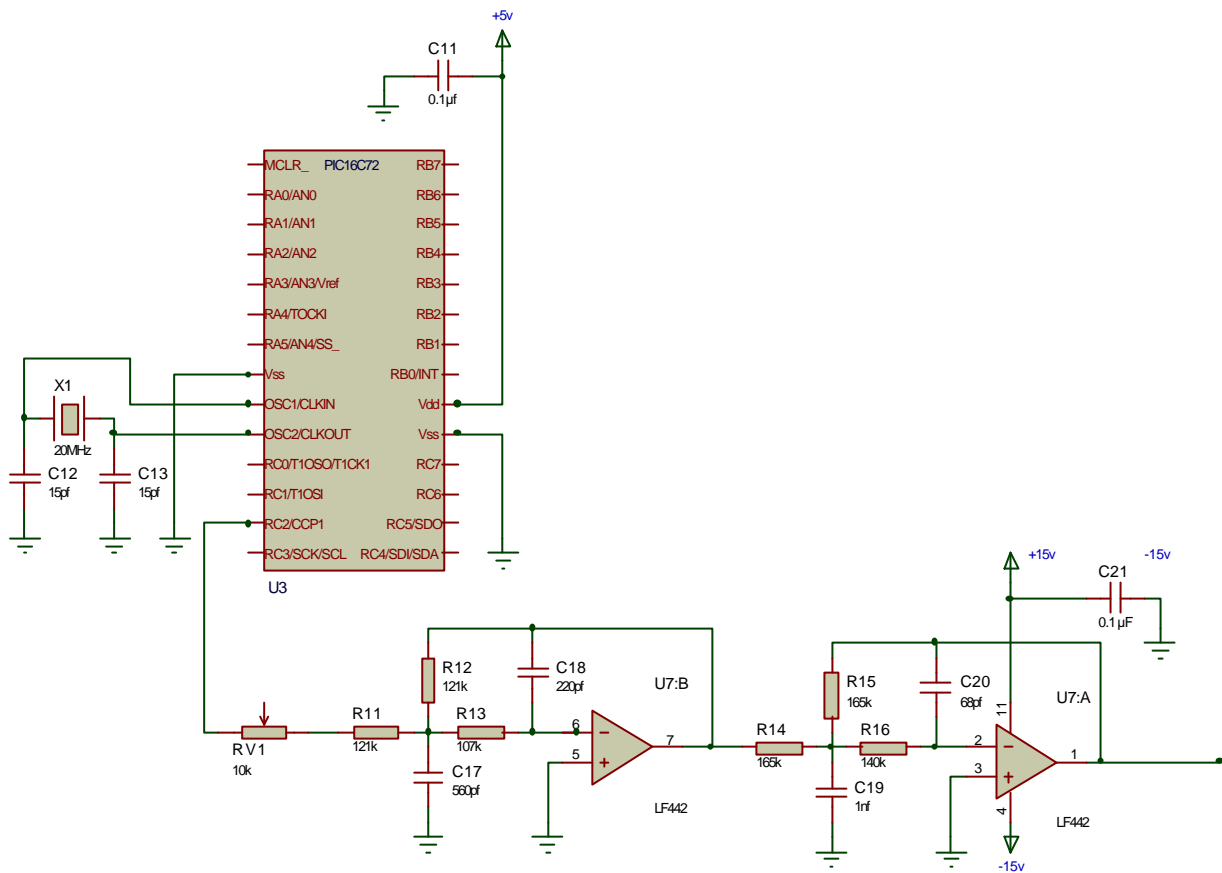


Figure 33 : Schéma électrique du filtre passe bas du module PWM.

RC2/CCP1 représente la sortie du module PWM.

- ✓ La courbe de gain théorique et pratique du filtre ($20 \log \frac{V_s}{V_e}$) se trouvent en annexe F.
- ✓ La documentation technique du PIC16C72 est disponible sur le CD-ROM :
 Répertoire : Doc
 Référence : PIC16C72.pdf

V.4.6 – Etude de l'amplificateur audio

Cet amplificateur est réalisé à partir d'un amplificateur audio de puissance LM386 de National Semiconductor. Le gain minimal interne de ce circuit atteint 20dB.

Schéma électrique :

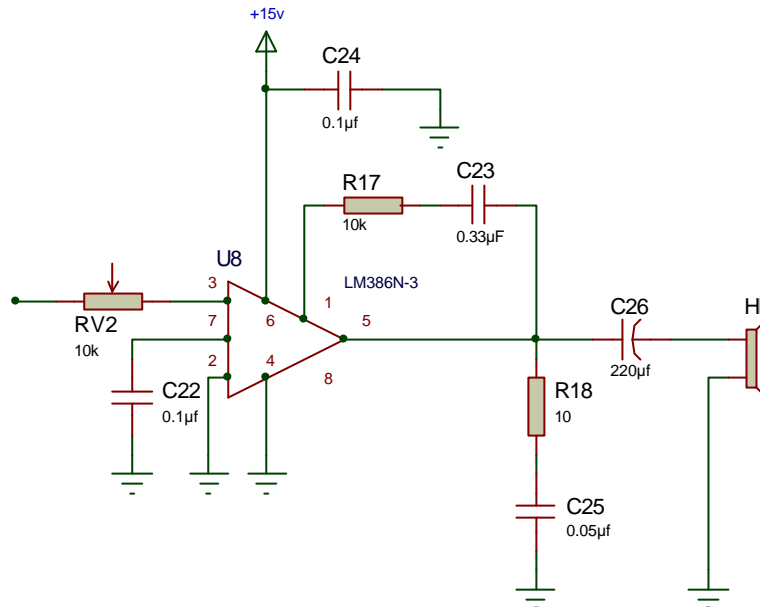


Figure 36 : Schéma électrique de l'amplificateur audio.

La résistance R17 en série avec le condensateur C23 permettent d'amplifier les basses fréquences.

- ✓ Les résultats des tests du gain de l'amplificateur se trouvent en annexe G.
- ✓ La documentation technique du LM386 est disponible sur le CD-ROM :
Répertoire : Doc
Référence : LM386.pdf

V.4.7 – Etude de l'alimentation

Les composants utilisés dans cette application ne fonctionnent pas tous avec la même valeur de tension d'alimentation. En effet, le PIC et la RAM utilisent le +5v tandis que les filtres et amplificateurs audio utilisent le +15v/-15v. A l'aide de régulateurs de tension, il est possible de fabriquer ces valeurs à partir d'une seule alimentation stabilisée, comme le montre le schéma suivant.

Schéma électrique :

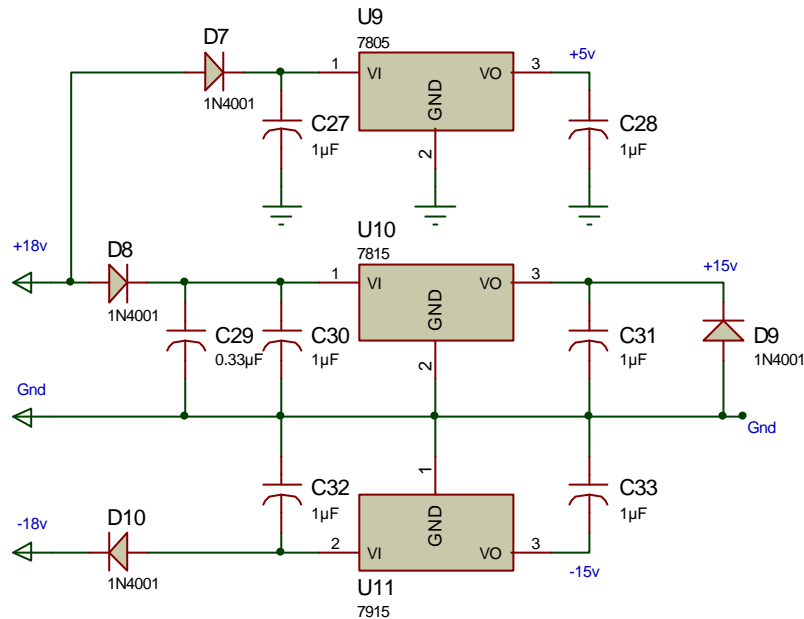


Figure 37 : Schéma électrique de l'alimentation.

Les trois régulateurs sont montés en parallèle et alimentés par une alimentation stabilisée. En sortie du régulateur MC7805CT, nous obtenons le +5v, le +15v avec le MC7815CT et le -15v avec le régulateur MC7915CT. Des condensateurs de découplage ajoutés aux entrées et sorties des régulateurs permettent d'augmenter la stabilité de la régulation. Ils servent aussi de filtres.

Pour être sûr d'obtenir le +15v/-15v, l'alimentation stabilisée doit pouvoir délivrer une tension un peu supérieure à ces valeurs. Par exemple, le +18v/-18v convient parfaitement.

Un interrupteur situé entre l'alimentation et les diodes D7, D8, D10 peut être ajouté pour faire office d'interrupteur général.

V.4.8 – Schéma électrique complet du circuit

La figure suivante montre le schéma électrique complet du circuit.

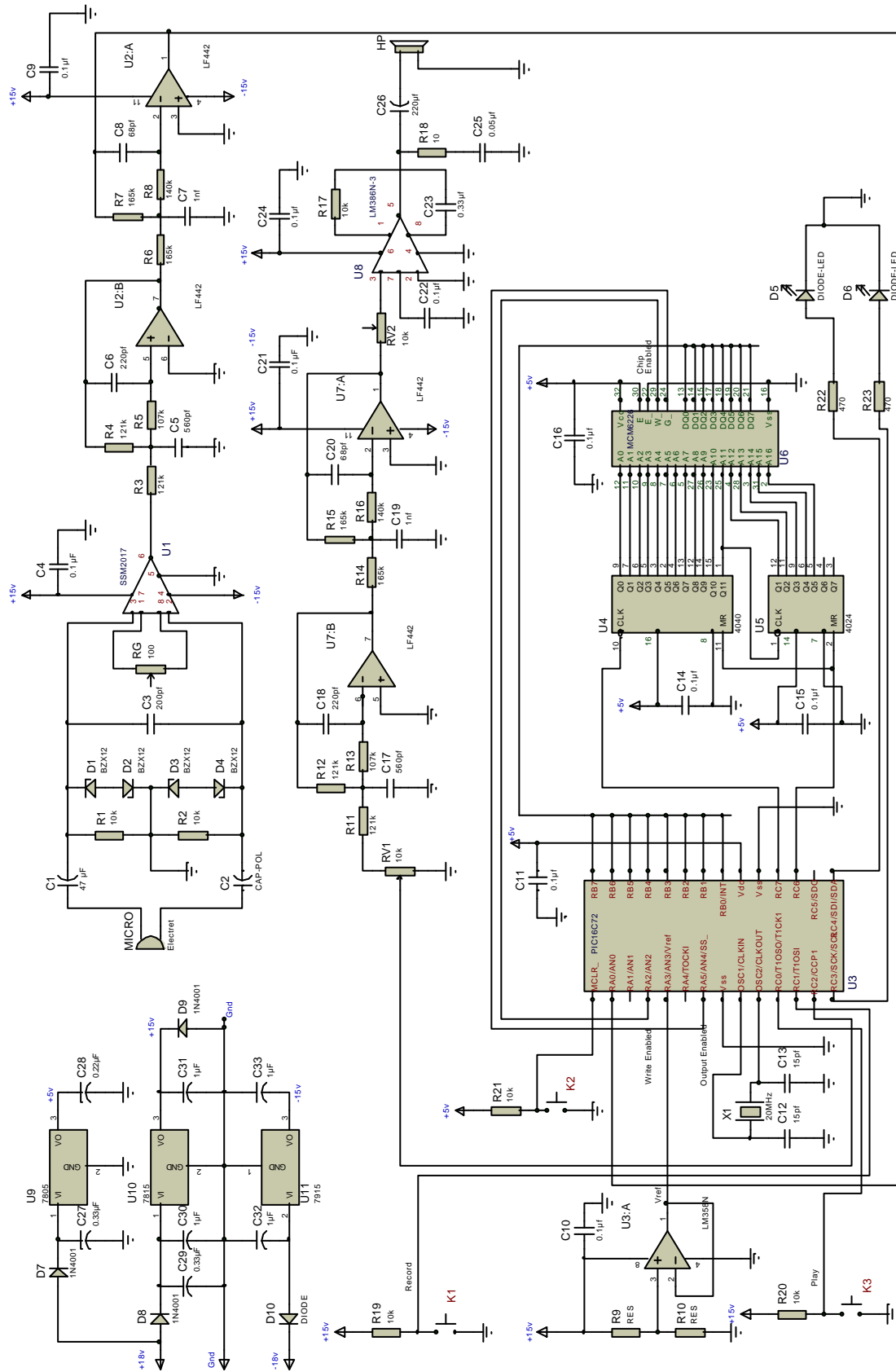


Figure 38 : Schéma électrique complet du circuit ADPCM.

V.5 – Réalisation du circuit

Après routage du circuit précédent à l'aide du logiciel Ares de Labcenter Electronics, nous obtenons le typon suivant (échelle réduite) :

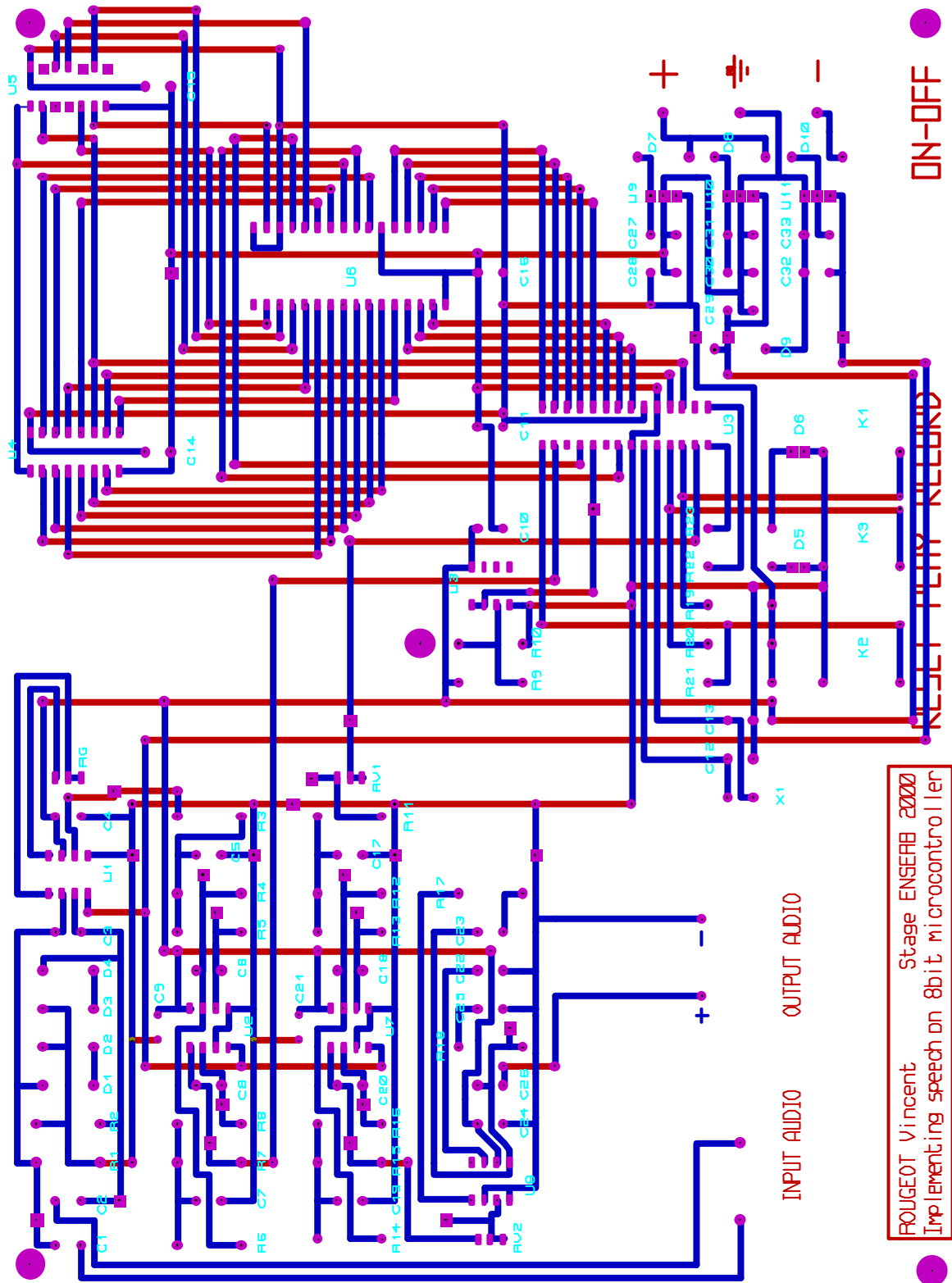


Figure 39 : Typon du circuit ADPCM.

✓ Les typons pour la réalisation du circuit ADPCM sont disponibles en annexe H.

Quantité	Désignation	Caractéristiques	Prix TTC Unitaire	Prix TTC Total
6	C27, C28, C30, C31, C32, C33	Condensateur 1 µF électrochimique	1.90	11.40
2	C29, C23	Condensateur 0.33 µF électrochimique	3.46	6.92
2	C12, C13	Condensateur 15 pF céramique	0.53	1.06
10	C4, C9, C10, C11, C14, C15, C16, C21, C22, C24	Condensateur 0.1 µF polyester	0.65	6.50
2	C1, C2	Condensateur 47 µf Tantale 63v	8.10	16.20
3	C3, C6, C18	Condensateur 220 pF céramique	0.60	1.80
2	C5, C17	Condensateur 560 pF céramique	0.54	1.08
2	C7, C19	Condensateur 1 nF céramique	0.90	1.80
2	C8, C20	Condensateur 68 pF céramique	2.00	4.00
1	C25	Condensateur 5 nF céramique	0.90	0.90
1	C25	Condensateur 220 µF électrochimique	2.16	2.16
6	R1, R2, R17, R19, R20, R21	Résistance carbone 10 KΩ ¼ W	0.15	0.90
4	R3, R4, R11, R12	Résistance carbone 121 KΩ ¼ W	0.15	0.60
4	R6, R7, R14, R15	Résistance carbone 165 KΩ ¼ W	0.15	0.60
2	R5, R13	Résistance carbone 107 KΩ ¼ W	0.15	0.30
2	R8, R16	Résistance carbone 140 KΩ ¼ W	0.15	0.30
1	R18	Résistance carbone 10 Ω ¼ W	0.15	0.15
2	R22, R23	Résistance carbone 470 Ω ¼ W	0.15	0.30
1	R9	Résistance carbone 200 Ω ¼ W	0.15	0.15
1	R10	Résistance carbone 1 KΩ ¼ W	0.15	0.15
2	D5, D6	DEL rouge Ø5mm	1.80	3.60
4	D1, D2, D3, D4	Diode zener BZX12	0.60	2.40
4	D7, D8, D9, D10	Diode 1N4007	0.58	2.32
2	RV1, RV2	Trimer 10 KΩ	1.98	3.96
1	RG	Trimer 100 Ω	1.98	1.98
1	X1	Quartz 20 MHz	30.00	30.00
1	U11	Régulateur MC7915CT	2.10	2.10
1	U9	Régulateur MC7805CT	8.38	8.38
1	U10	Régulateur MC7812CT	8.38	8.38
1	U1	Préamplificateur SSM2017	20.60	20.60
2	U2, U7	AOP LF442 JFET	10.92	21.84
1	U8	Amplificateur audio LM386	5.20	5.20
1	U3	AOP LM358N	4.75	4.75
1	U4	Compteur binaire CD4040B	45.40	45.40
1	U5	Compteur binaire CD4024B	39.50	39.50
1	U6	SRAM TC551001 128 Ko	87.50	87.50
1	U3	PIC 16C72 JW	116.80	116.80
3	K1, K2, K3	Bouton poussoir	27.54	82.62
5		Support DIP 8	2.68	13.40
1		Support DIP 14	4.84	4.84
1		Support DIP 48	15.50	15.50
1		Support DIP 24	11.59	11.59
1		Support DIP 16	5.83	5.83
1		Interrupteur 2 états 2 voix	3.00	3.00
7		Fiche banane + châssis	4.00	28.00
1		Plaque Epoxy 200x300 double faces	110.65	110.65
PRIX TOTAL TTC en Francs			742.41	

Référence prix : Radiospares

Figure 40 : Nomenclature du circuit ADPCM.

V.6 – Le codage ADPCM

ADPCM pour Adaptive Differential Pulse-Code Modulation signifie Modulation d'Impulsion Différentielle Adaptative. Ce codage appartient à la famille des algorithmes de compression/décompression de la parole. Il a pour but de convertir sur 4 bits des échantillons de voix codés sur 16 bits en PCM.

Cette méthode de compression tire son avantage du fait qu'il existe d'importantes corrélations entre les échantillons consécutifs de la parole. C'est à dire qu'il est possible, à partir d'un algorithme simple, de prédire ce que sera le prochain échantillon à partir de l'échantillon précédent.

Les tests montrent que l'erreur, résultant de la comparaison de l'échantillon prédit à l'échantillon réel est très faible.

Pour obtenir la compression de donnée, il y a codage de la différence entre la valeur échantillonnée et la valeur prévue par l'algorithme. Le fait de coder cette différence est suffisant en termes de qualité pour tout ce qui est phonie. De plus, cette méthode permet de gagner de la place lors du stockage des données puisque chaque échantillon est divisé par 4. Elle préserve aussi de manière convenable la qualité du signal de la parole.

V.5.1 – Quel algorithme utiliser ?

Dans la pratique, il existe 3 algorithmes :

- Microsoft (MS ADPCM),
- Creative Labs (Creative ADPCM),
- Interactive Multimedia Association's ADPCM (IMA ADPCM).

L'algorithme utilisé dans cette application est basé sur une version de l'IMA ADPCM développé par la société Intel/DVI. Cet algorithme a la particularité de n'utiliser que des fonctions mathématiques simples qu'un microcontrôleur 8 bits est à même d'exécuter. Cet algorithme utilise, pour la compression et la décompression, une quantification adaptative utilisant des prédictions fixes. Ces prédictions étant réalisées à partir de tables développées par Intel/DVI.

V.5.2 – Compression / Décompression

V.5.2.1 – Compression

La figure 41 représente le schéma bloc de la méthode de compression ADPCM utilisée dans l'application.

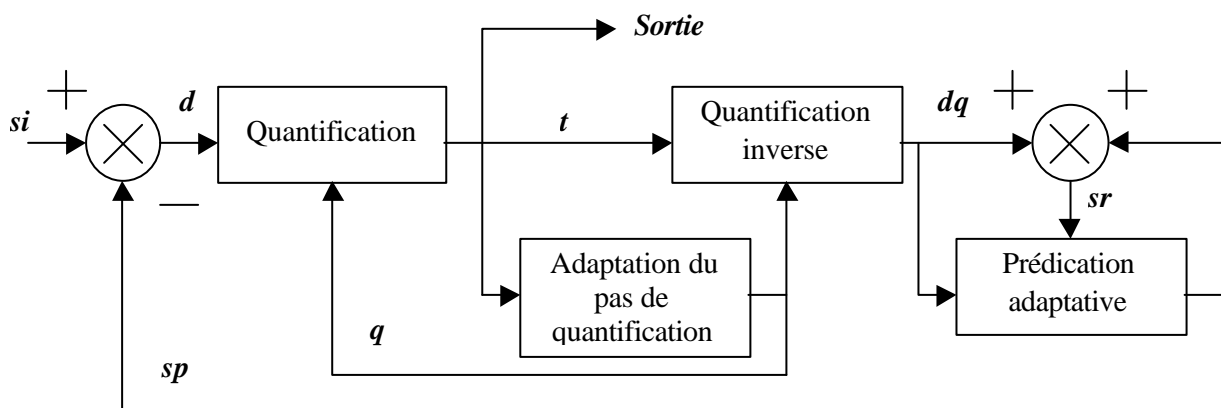


Figure 41 : Schéma bloc de l'encodage ADPCM.

L'échantillon sp prédit lors du précédent encodage via $si-1$, $d-1$, $q-1$ et $t-1$ est soustrait à l'échantillon d'entrée si pour produire la différence d . Le bloc de quantification traite la donnée d et la transforme en t , un code ADPCM de 4 bits. Ce code t sera traité via le bloc de prédiction de façon à prédire l'échantillon suivant $sp+1$ qui sera de nouveau soustrait par l'échantillon d'entrée $si+1$ pour produire $d+1$ et $t+1$. Et ainsi de suite.

V.5.2.2 – Décompression

La figure 42 représente le schéma bloc de la méthode de décompression utilisée dans l'application.

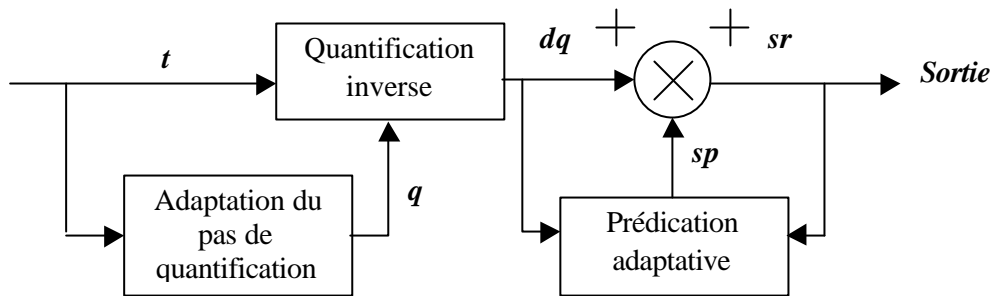


Figure 42 : Schéma bloc du décodage ADPCM.

Le code ADPCM de 4 bits, t , est traité par la quantification inverse, lequel produit par la différence dq . La différence dq est additionnée à l'échantillon prédit $sp-1$ pour produire l'échantillon codé sur 16 bits de sortie, sr . L'échantillon sr est alors sauvé dans sp pour être utilisé lors de la prochaine itération du décodeur.

V.5.3 – Au niveau informatique

Au niveau informatique, la décompression et la compression se traduisent par deux fonctions.

- **ADPCMEncoder (input, output)**
Le paramètre **input** contient le signal non codé.
Le paramètre **output** contient le signal compressé.
- **ADPCMDecoder (input, output)**
Le paramètre **input** contient le signal à décompresser.
Le paramètre **output** contient le signal décompressé.

Ces deux fonctions développées par l'IMA et Intel/DVI sont distribués gratuitement sur Internet et directement exploitable.

- ✓ La fonction *ADPCMEncoder ()* et *ADPCMDecoder ()* sont disponibles sur le CD-ROM :
Répertoire : An643/sources
Référence : *adpcm.c*

V.7 – Etude logicielle

Après avoir vu la partie matérielle de l'application, nous devons maintenant nous intéresser à la partie logicielle.

V.7.1 – Choix du langage de programmation

Comme nous l'avons vu dans le chapitre III, il existe plusieurs langages de programmation, plus ou moins évolués, qui permettent d'obtenir, via un compilateur adéquate, un fichier binaire exploitable par les microcontrôleurs.

Bien que nous bénéficions du logiciel MPLAB de Microchip pour programmer en assembleur, le langage C est choisi comme langage de programmation. Et ceci, pour deux raisons : la première, les fonctions d'encodage et de décodage ADPCM de l'IMA ADPCM sont écrites en C; la seconde, le langage C est bien plus intuitif et plus simple à utiliser que l'assembleur.

V.7.2 – Choix du compilateur C

Pour programmer en C, il faut trouver un compilateur C qui puisse créer des fichiers binaires exploitables par les microcontrôleurs PIC. Le choix du compilateur s'est porté sur le logiciel HI-TECH C Compiler de la société HI-TECH Software. Celui-ci fonctionne sous environnement DOS ou commandes MS-DOS de Windows.

Bien qu'il fonctionne sous DOS, il possède une interface graphique sobre et sa prise en main est très aisée.

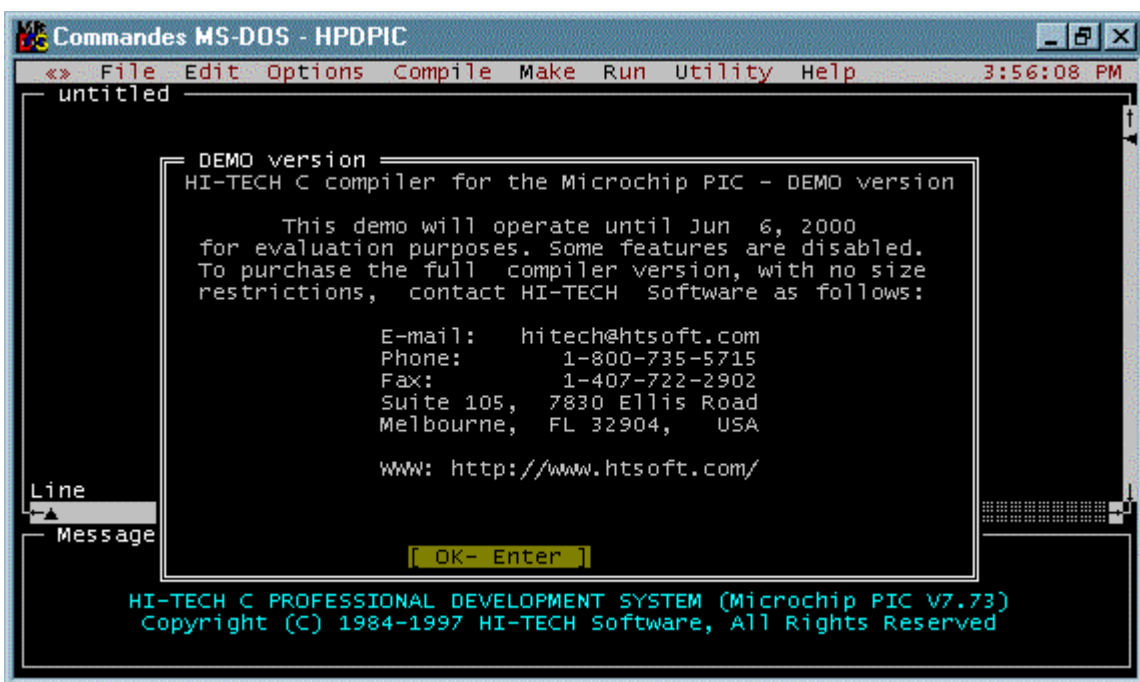
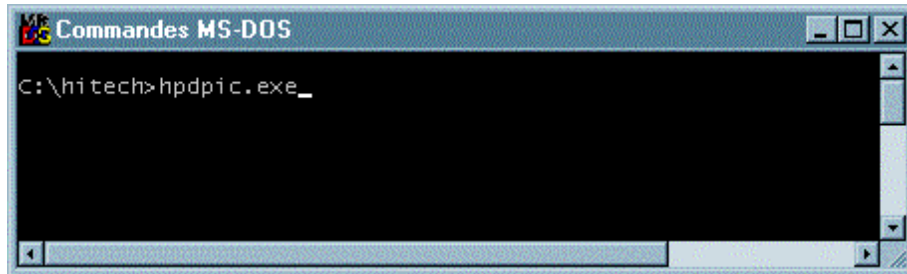


Figure 43 : Compilateur HI-TECH C Compiler de HI-TECH Software.

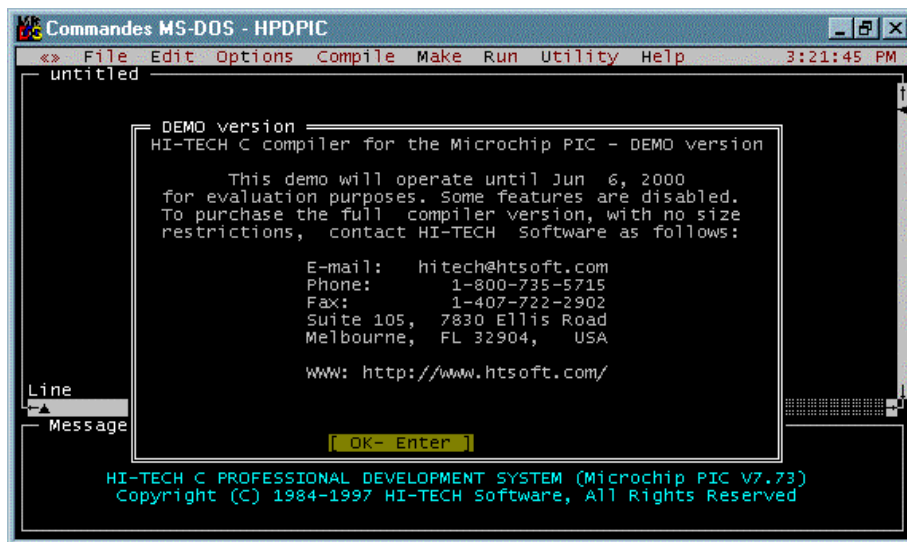
V.7.3 – Fonctionnement du compilateur HI-TECH C Compiler

Son fonctionnement est très simple et se résume à ces quelques étapes :

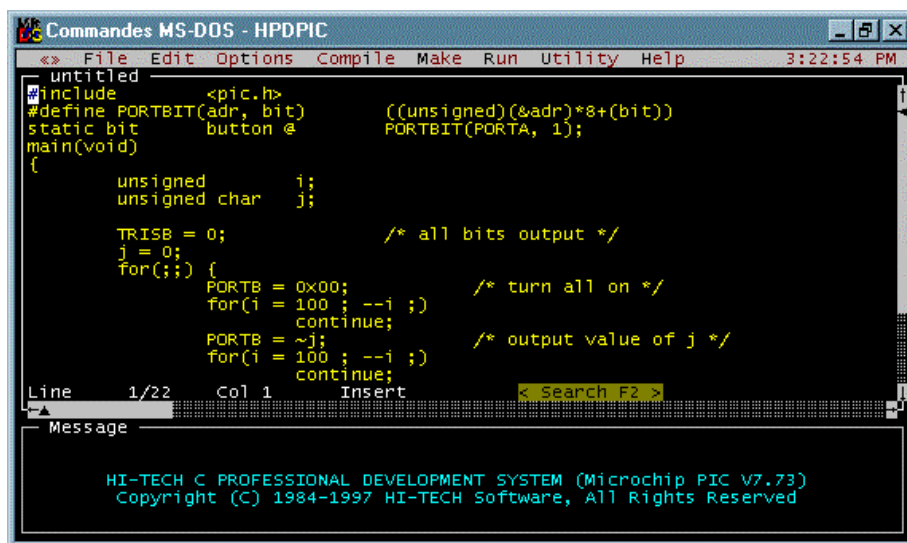
- Lancer le programme **hdpic.exe**.



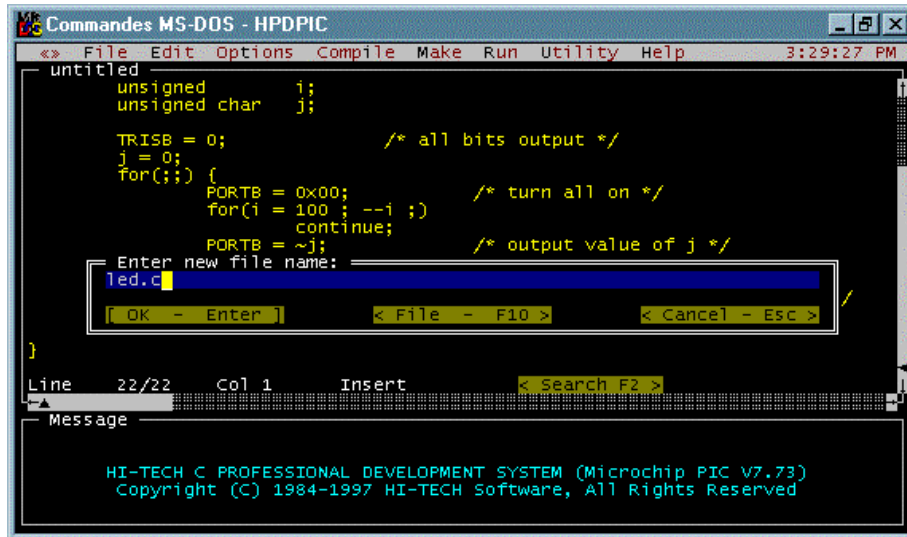
- Lorsque l'écran suivant apparaît, cliquer sur **Entrée**.



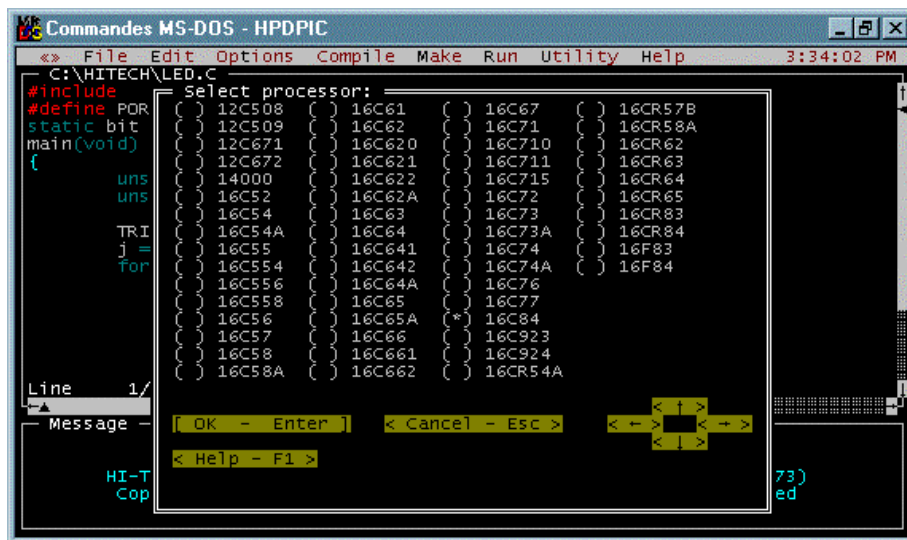
- Cliquer sur **File** puis **New**.
- Taper le programme en C.



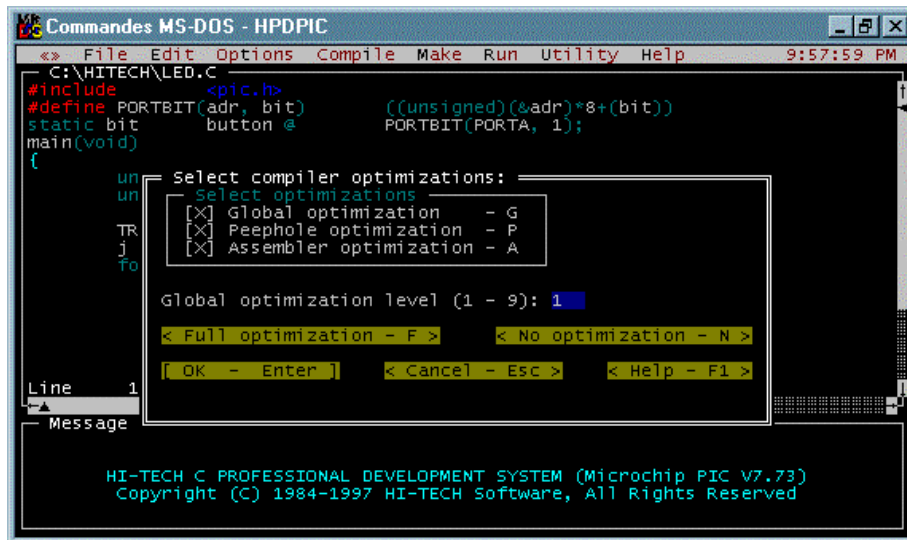
- Enregistrer le programme : cliquer sur **File** puis **Save as...**
- Taper le nom du fichier dans la fenêtre qui apparaît (ici **led.c**).
- Cliquer sur **OK**.



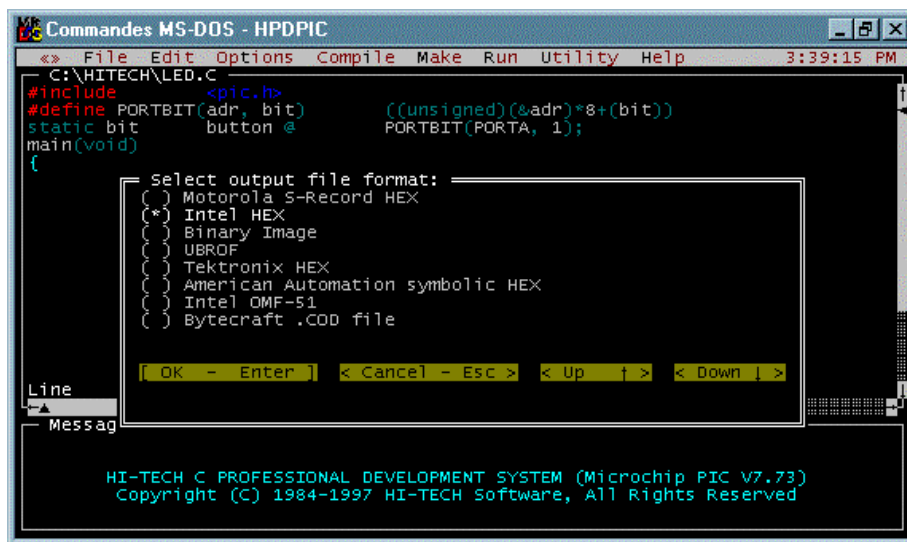
- Pour compiler le programme : cliquer sur **Compile** puis **Compile and link**.
- Choisir le type de microcontrôleur (ici **PIC16C84**).
- Cliquer sur **OK**.



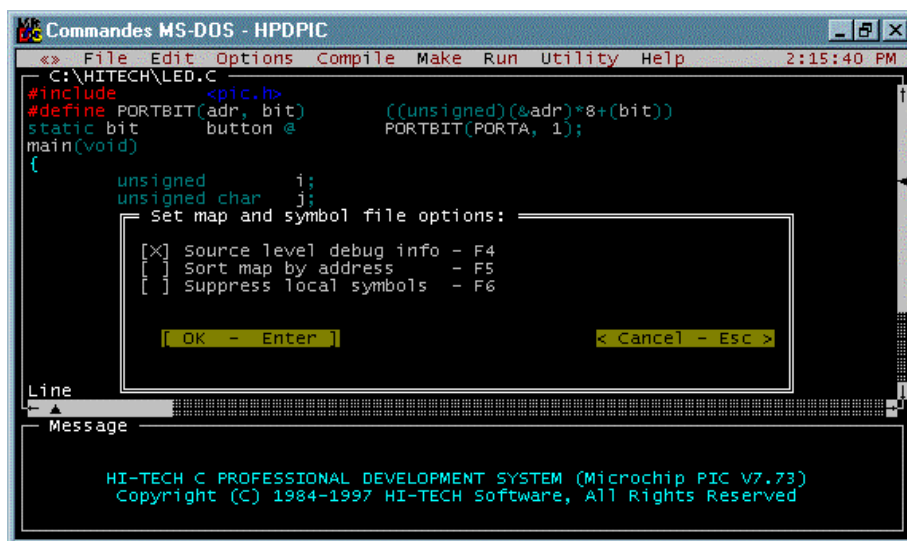
- Cliquer sur **< Full optimization - F >** puis **OK**.



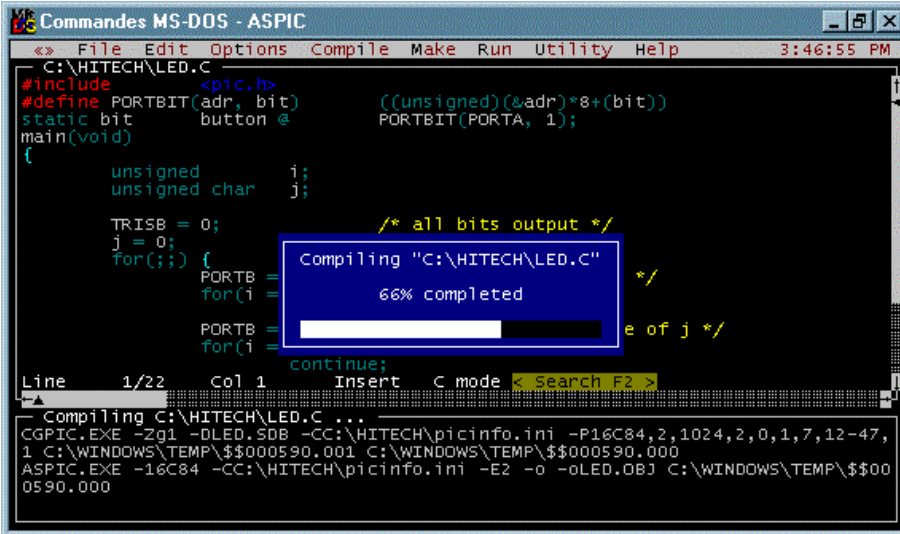
- Choisir le format du fichier binaire (ici **Intel HEX**). Cliquer sur **OK**.



- Sélectionner **Source level debug info - F4**. Cliquer sur **OK**.



- ✓ Attendre que la compilation se termine.



```
Commandes MS-DOS - ASPIC
C:\HITECH\LED.C
File Edit Options Compile Make Run Utility Help 3:46:55 PM
#include <pic.h>
#define PORTBIT(adre, bit) ((unsigned)&adre)*8+(bit)
static bit button e PORTBIT(PORTA, 1);
main(void)
{
    unsigned char i;
    unsigned char j;

    TRISB = 0; /* all bits output */
    j = 0;
    for(;;) {
        PORTB =
        for(i =
        PORTB =
        for(i =
        continue;
Line 1/22 Col 1 Insert C mode Search F2
Compiling C:\HITECH\LED.C ...
CGPIC.EXE -Zg1 -DLED.SDB -CC:\HITECH\picinfo.ini -P16C84,2,1024,2,0,1,7,12-47,
1 C:\WINDOWS\TEMP\$$000590.001 C:\WINDOWS\TEMP\$$000590.000
ASPIC.EXE -16C84 -CC:\HITECH\picinfo.ini -E2 -o -oLED.OBJ C:\WINDOWS\TEMP\$$00
0590.000
```

- Si aucune erreur n'est détectée, le message **Compilation successful** apparaît dans la fenêtre basse du compilateur. Le fichier binaire est alors disponible dans le même répertoire que le fichier source (ici **led.hex** à partir de **led.c**).
- ✓ Une version d'évaluation de *HI-TECH C Compiler* est disponible sur le CD-ROM :
Répertoire : *Hitech*
Référence : *hdpic.exe*

V.8 – Développement du programme du circuit ADPCM

D'après tout les éléments vu précédemment nous aboutissons à l'organigramme suivant :

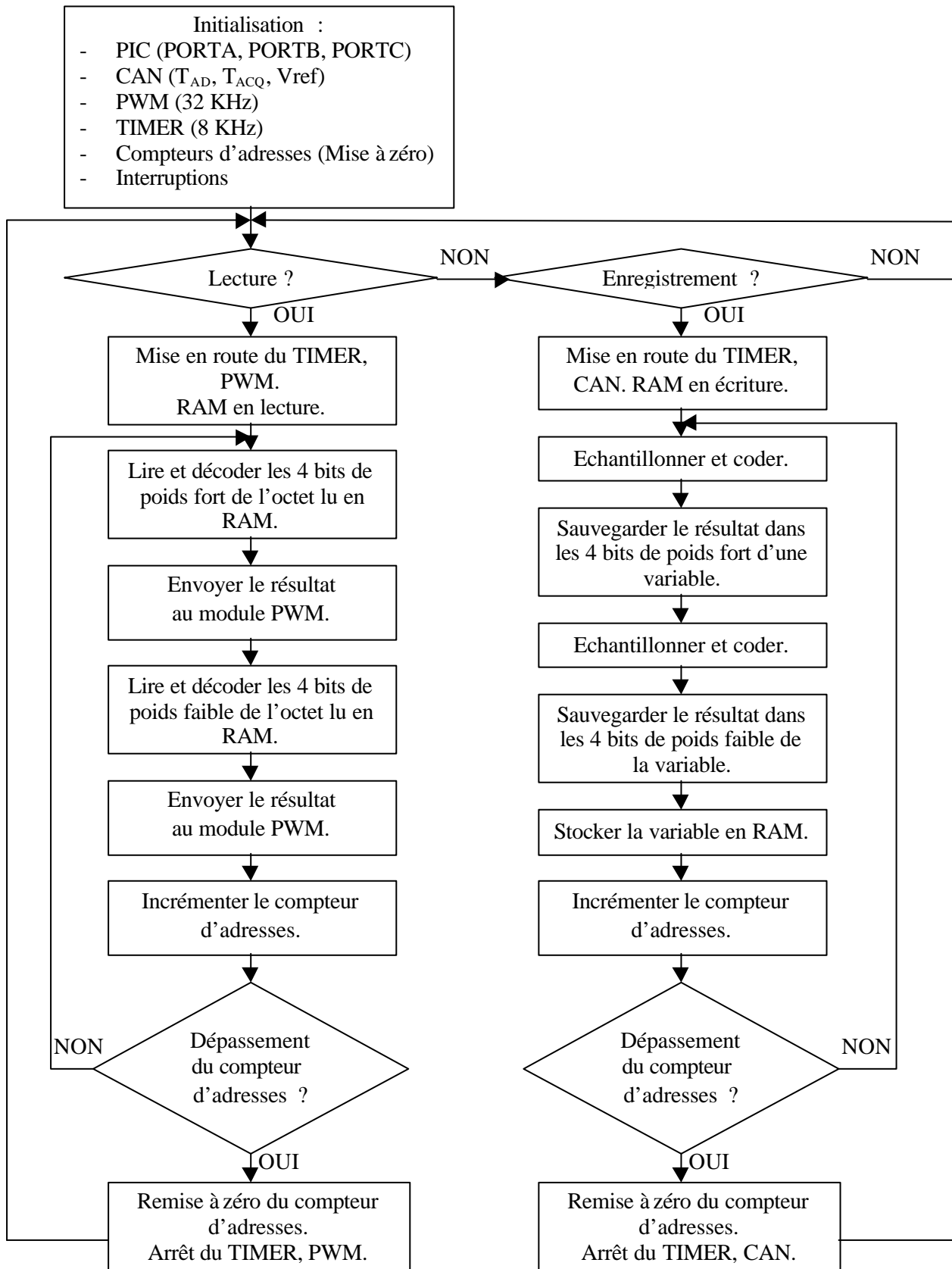


Figure 44 : Organigramme du programme du circuit ADPCM.

A partir de cet organigramme, le programme en langage C est réalisé. Celui-ci se compose de quatre fichiers dépendant les uns et des autres. Ceux-ci s'intitulent **declare.c**, **adpcm.c**, **pic1672.h**, et **projet.c**.

- **declare.c** : ce fichier regroupe les déclarations des fonctions utilisées dans **projet.c**.
- **adpcm.c** : ce fichier regroupe les deux fonctions de codage et de décodage ADPCM mises au point par l'IMA et développées par Intel/DVI : ADPCMEncoder () et ADPCMDecoder ().
- **pic1672.h** : ce fichier regroupe tout les noms symboliques attribués à chacun des registres du microcontrôleur PIC 16C72. Ce fichier est distribué avec le logiciel HI-TECH C Compiler.
- **projet.c** : ce fichier regroupe les fonctions et le programme principal de l'application.

✓ *Les listings des fichiers declare.c, adpcm.c, pic1672.h et projet.c sont disponibles en annexe I.*

Après la compilation de ces quatre fichiers à l'aide du logiciel HI-TECH C Compiler, nous obtenons le fichier **projet.hex** qui n'est autre que le fichier binaire à implanter dans le microcontrôleur pour faire fonctionner le circuit ADPCM.

Ce projet m'a permis de découvrir l'univers des microcontrôleurs PIC, tant dans leur théorie que dans leur développement. En effet, j'ai étudié, programmé et réalisé des applications fonctionnant avec ces microcontrôleurs : trois programmeurs de PIC 16F84 et un circuit de traitement de la voix utilisant le codage ADPCM. Ce dernier montage met en évidence le fait que les microcontrôleurs actuels peuvent désormais se substituer, dans certaines applications de traitement du signal, au processeur DSP.

Lors de mes recherches, j'ai pu constater l'importance et la précision des publications dans les revues spécialisées ainsi que sur les sites Internet accessibles à tout public ; ce qui laisse présager un bel avenir pour les PIC, tant dans l'industrie qu'auprès des amateurs avertis.

Bibliographie

Les microcontrôleurs PIC : description & mise en œuvre
Auteur Christian TAVERNIER
Edition Dunod

La mise en œuvre des microcontrôleurs : MICROCHIP PIC16Cxx
Auteur Martina Kost

Documentation au format HTML et logiciels des programmeur PARPIC et QUICK AND DIRTY :
<http://www.nexuscomputing.com/~picarchive/icp84.html>

Documentation au format PDF et logiciel du programmeur PICPROG 2000
<http://www.supelec-rennes.com/ren/perso/jweiss/pic/pic.htm> (picprog2000.pdf)
<http://www.supelec-rennes.com/ren/perso/jweiss/tools/picprogjw/picprogjw.html> (picprogjw.zip)

Documentation technique du PIC 16F84 au format PDF.
<http://www.microchip.com/10/Lit/PICmicro/16F8X/30430/index1.htm> (30430.pdf)

Documentation technique du PIC 16C72 au format PDF.
<http://www.microchip.com/10/Lit/PICmicro/16C7X/39016/index.htm> (39016a.pdf)

Note d'application AN643 sur l'ADPCM format PDF.
<http://www.microchip.com/10/Appnote/Category/PIC16/00643/index6.htm> (00643a.pdf)
<http://www.microchip.com/10/Appnote/Category/PIC16/00643/idxZIP6.htm> (00643.zip)

Note d'application AN546 sur l'utilisation du CAN des PIC au format PDF.
<http://www.microchip.com/10/Appnote/Category/16CXX/00546/index1.htm> (00546e.pdf)
<http://www.microchip.com/10/Lit/PICmicro/16F8X/00546/idxZIP1.htm> (00546.zip)

Logiciel MPLAB 5.00 et sa documentation
<http://www.microchip.com/10/Tools/PICmicro/DevEnv/MPLABi/Software/v50000/index.htm>

Documentation technique du préamplificateur SS2017.
http://www.analog.com/pdf/1782_c.pdf

Documentation technique de l'amplificateur opérationnel LF442.
<http://www.national.com/pf/LF/LF442.html>

Documentation technique de l'amplificateur audio LM386.
<http://www.national.com/pf/LM/LM386.html>

Documentation technique de l'amplificateur opérationnel LM358.
<http://www.national.com/pf/LM/LM358.html>

Documentation technique de la SRAM TC551001.
<http://www.toshiba.com/taec/components/Datasheet/1001c.pdf>

Documentation technique des compteurs binaires CD40XXB.
<http://www.ti.com/sc/docs/products/logic/cd4040b.htm> (schs0.30.pdf)

Annexe A : Typons du programmeur PARPIC.

Annexe B : Typons du programmeur QUICK AND DIRTY.

Annexe C : Typons du programmeur PICPROG 2000.

Annexe D : Article "Implementing speech on 8bit microcontroller" du magazine Embedded Systems.

Annexe E : Test du gain du préamplificateur SSM2017.

Annexe F : Test du gain du filtre passe bas du 4^{ième} ordre de type Butterworth.

Annexe G : Test du gain de l'amplificateur audio LM386.

Annexe H : Typons du circuit de l'application ADPCM.

Annexe I : Listing des programmes de l'application ADPCM.