SOMMAIRE

CHAPITRE V CARTE DE TRANSMISSION

CHAPITRE VI LOGICIEL D'ACQUISITION DE DONNEES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES SIGLES ET ABREVIATIONS

ASCII	American Standard Code for Information Interchange
CCITT	Comité Consultatif International de Télégraphie et Téléphonie
CP	Clock Pulse
CS	Chip Select
CTS	Clear To Send
DCB	Device Control Block
DCE	Data Communication Equipement
DLL	Data Link Library
DLL	Divisor Latch Least significant bit
DLM	Divisor Latch Most significant bit
DSR	Data Set Ready
DTE	Data Terminal Equipement
DTR	Data Terminal Ready
EEPROM	Electrically Erasable Programmable Read Only Memory
EIA	Electronic Industries Association
EPD	EPROM Data
EPROM	Erasable Programmable Read Only Memory
IER	Interupt Enable Register
LCR	Line Control Register
LED	Light Emmiting Diod
LSB	Least Significant Bit
MCR	Modem Control Register
MSB	Most Significant Bit
PROM	Programmable Read Only Memory
R/W	Read Write
RAM	Random Acces Memory
RAZ	Remise à Zéro
RM	Rapidité de Modulation
ROM	Read Only Memory
RS232C	Recommended Standard number 232 revision C
RTS	Request To Send
TTL	Transistor Transistor Logic
UART	Universal Asynchronous Receiver Transmitter
VLSI	Very Large Scale Integration

Introduction

Actuellement, les procédés d'intégration à grande échelle (VLSI), utilisés dans la fabrication des microprocesseurs et particulièrement des circuits mémoires comme les EPROMs sont très performants.

D'autre part, les nouvelles technologies, comprenant d'un côté les outils informatiques et d'un autre côté l'électronique digitale sont devenues de plus en plus puissantes et dominent presque dans la totalité des applications modernes.

En conséquence, la plupart des appareils que nous utilisons quotidiennement sont équipées d'une carte micro-programmée où sont mémorisées des instructions et des données nécessaires à leur fonctionnement.

En géophysique, outre la puissance, la facilité et les qualités des traitements et des interprétations effectuées sur micro-ordinateur, il y a nécessité d'utiliser des dispositifs de plus en plus performants et compétitifs pour obtenir les meilleurs résultats et le plus de données possibles au cours d'une campagne de mesures.

Ce qui fait que la majeure partie des appareils utilisés à l'Institut et Observatoire de Géophysique d'Antananarivo à nos jours a été conçue selon cette nouvelle technologie. Le résistivimètre SYSCAL R2 et le TERRALOC MK6 sont des exemples typiques de ces matériels.

Mais avec le temps, l'appareil s'affaiblit et une partie ou la totalité de l'information gravée dans les mémoires (généralement des EPROMs) peuvent se perdre.

Face à ce problème, mon collègue NJARASOATSINJOAVO Tsitohaina et moi même ont réalisé un programmateur d'EPROM pour pouvoir recharger les contenus d'un EPROM défectueux dans un EPROM vierge.

Notons que la programmation d'une mémoire peut être faite manuellement. Cependant, cette tâche serait pénible et demande plusieurs minutes ou même quelques heures de manipulation si le nombre de données à écrire est grand.

Au cours de cette réalisation, je me suis occupé de l'interfaçage du programmateur conçu par NJARASOATSINJOAVO Tsitohaina à l'ordinateur afin :

- de préserver les informations stockées dans un EPROM et de les sauvegarder sous forme de fichier avant qu'elles ne disparaissent ;
- d'automatiser et de minimiser le temps de programmation.

Le présent mémoire est composé de sept chapitres et s'articule comme suit :

- dans les deux premiers chapitres sont rappelées les notions utilisées
- dans le troisième chapitre, nous parlerons des matériels et des méthodes employés
- le quatrième, le cinquième et le sixième chapitre sont consacrés aux réalisations, comprenant d'une part les réalisations électroniques et d'autre part l'élaboration des programmes d'interfaçage.
- au terme de ce travail, nous présenterons les résultats obtenus ainsi que les discussions à propos de l'interface réalisée.

CHAPITRE I : NOTIONS SUR LES MEMOIRES

Tout d'abord, selon Larousse, en informatique le terme information désigne un élément de connaissance susceptible d'être codé pour être conservé, traité ou communiqué.

Appliquer ensemble les concepts mathématiques de l'algèbre de Boole et les lois de l'électronique au traitement des informations du monde réel nécessite deux notions fondamentales indissociables :

- premièrement, les informations à traiter doivent être représentées sous forme binaire ;
- deuxièmement, l'unité de traitement des données numérisées doit avoir une unité capable de maintenir l'état des bits et conserver les résultats intermédiaires pendant le traitement (Tisserant 2003), pour que la manipulation se déroule sans aucune perte d'information.

Pour assurer ces conservations, l'unité contiendrait au moins un ou plusieurs composants qui réalisent des fonctions de mémorisation appelées mémoires

I.1) Définition

Une mémoire est un dispositif qui peut être lue ou écrite, capable de stocker des informations ou des instructions et de les restituer à la demande du système qui la contient (Cattoen 2003).

I.2) Description

I.2.1) Vue interne

D'une façon simplifiée et vue de l'intérieur, une mémoire ressemble à une matrice ou tableau de m x 2^n éléments disposés suivant 2^n lignes et m colonnes (Tisserant 2003) ; chaque élément est formé par un registre ayant la capacité de stocker 1 bit appelé cellule mémoire ou point mémoire (Mercouroff 1990).

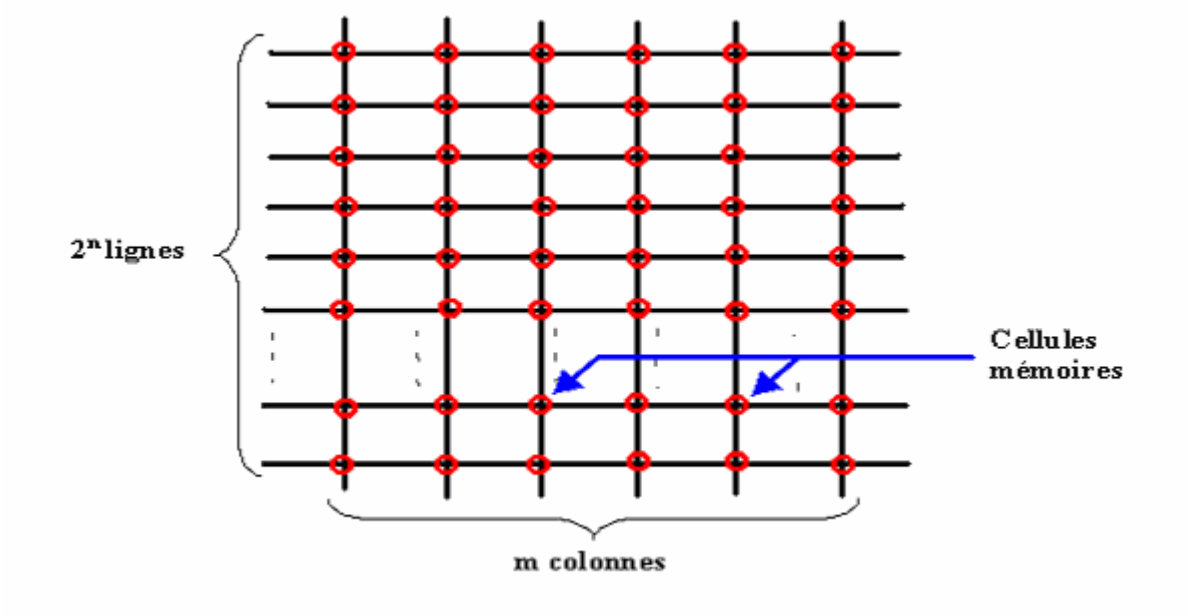


Figure I.1 Organisation d'un bloc mémoire en cellules

Les m cellules d'une ligne forment une case mémoire et constituent le mot de l'information.

Dans le cas où m=8, la capacité de la case serait 1 octet (Dowsing 1987). Ainsi, pendant une opération d'écriture ou une opération de lecture sur la mémoire, les m cellules sont traitées simultanément.

En d'autre terme, un bloc mémoire est un assemblage de m x 2^n cases, où chacune des cases est divisée en m unités de cellule mémoire de 1 bit.

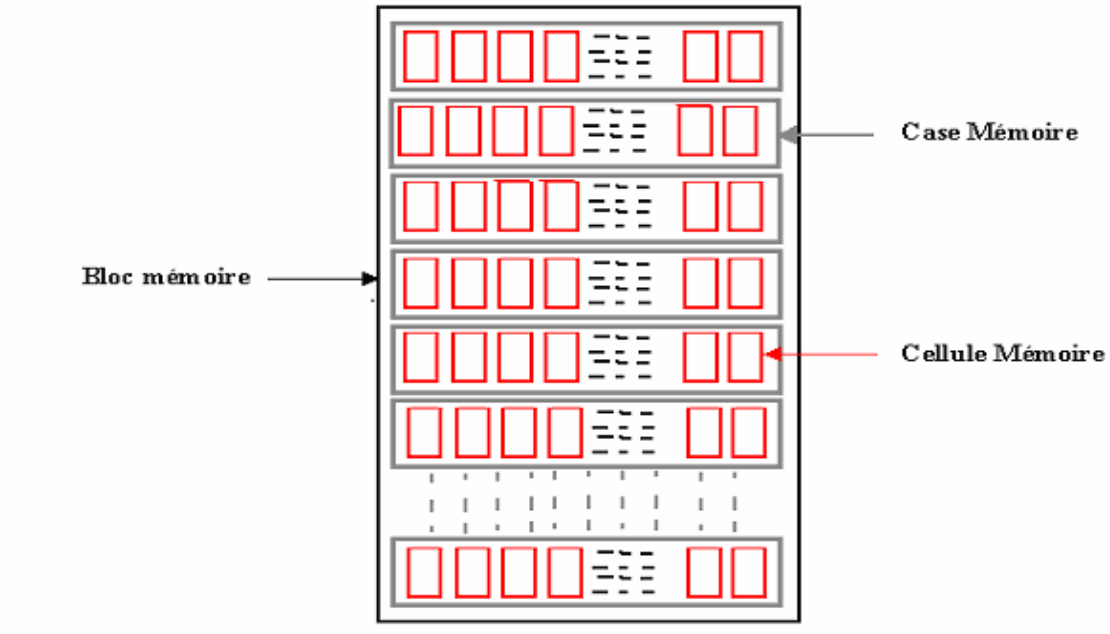


Figure I.2 Organisation d'un bloc mémoire en cases

I.2.2) Vue externe

Vu de l'extérieur, c'est un composant électronique multipôle dont les pattes sont divisées en quatre bus bien distincts (Tisserant 2003) :

- un bus de données composé de m lignes, servant à véhiculer les bits venant des circuits extérieurs vers les m cellules des cases ou inversement
- un bus d'adresse de n lignes de sens entrant, nécessaire à la localisation des cases
- une ligne de sélection CS pour valider la case sélectionnée
- une ligne de commande R/W permettant de choisir entre opération de lecture et opération d'écriture

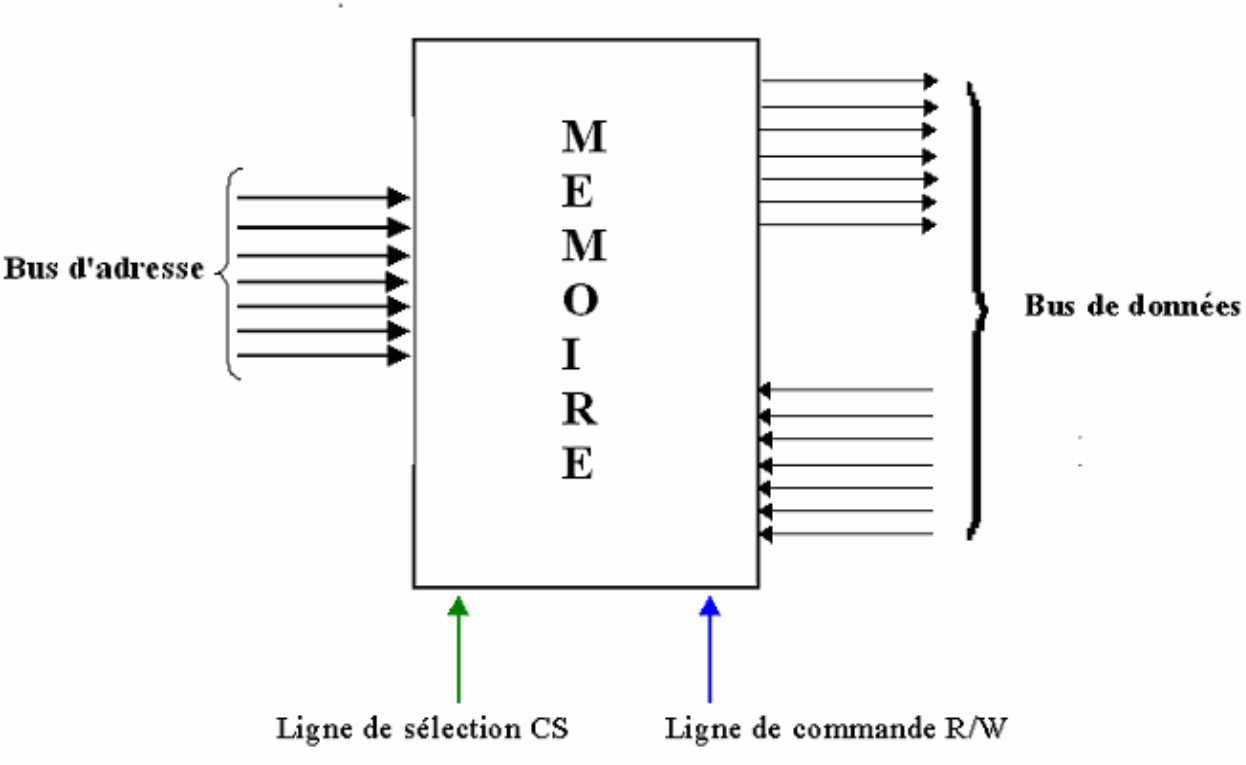


Figure I.3 Vue externe d'une mémoire

I.3) Caractéristiques d'une mémoire

En général, une mémoire se différencie d'une autre par un certain nombre de propriétés qualitatives et quantitatives, caractérisant sa taille, son accès, sa fréquence de fonctionnement, son débit et la permanence de ses données (Mercouroff 1990).

- **Capacité C d'une mémoire**

Cette grandeur représente deux aspects bien distincts.

- D'une part, la capacité exprime le nombre total de mots (exprimé en octet) que la mémoire peut contenir (Cattoen 2003).
- D'autre part, elle exprime le nombre de bits qui compose le mot de la mémoire, appelé aussi format par Tisserant (Tisserant 2003).

Remarques

1) La capacité en bits d'une mémoire est identique au nombre total de cellules de la mémoire

2) La capacité C en octet est exprimée en puissance de 2 et multiple de 1024 ou kilo.

Le tableau I.1 ci dessous résume les valeurs correspondantes aux autres préfixes les plus utilisés.

Tableau I.1 Les multiples de l'octet

SYMBOLE	PREFIXE	CAPACITE
1 K	Kilo	2^{10} = 1024
1 M	Méga	2^{20} = 1048576
1 G	Giga	2^{30} = 1073741824
1 T	Téra	2^{40} = 1099511627776

- **Accès à l'information**

Tout d'abord, le temps d'accès d'une mémoire est le temps nécessaire pour retrouver et lire le contenu d'une case correspondant à une adresse donnée (Tisserant 2003).

Si ce temps est indépendant de l'adresse de la zone à lire alors l'accès à l'information est dit aléatoire.

Par contre, s'il faut lire toutes les adresses du début jusqu'à l'adresse voulue alors l'accès est dit séquentiel.

- **Fréquence de fonctionnement d'une mémoire**

Par définition, la fréquence de fonctionnement d'une mémoire est inversement proportionnelle au temps de cycle T_c ; où un cycle (Figure I.4) est défini par le plus petit domaine possible compris entre deux requêtes successives faites au niveau de la mémoire (Tisserant 2003).

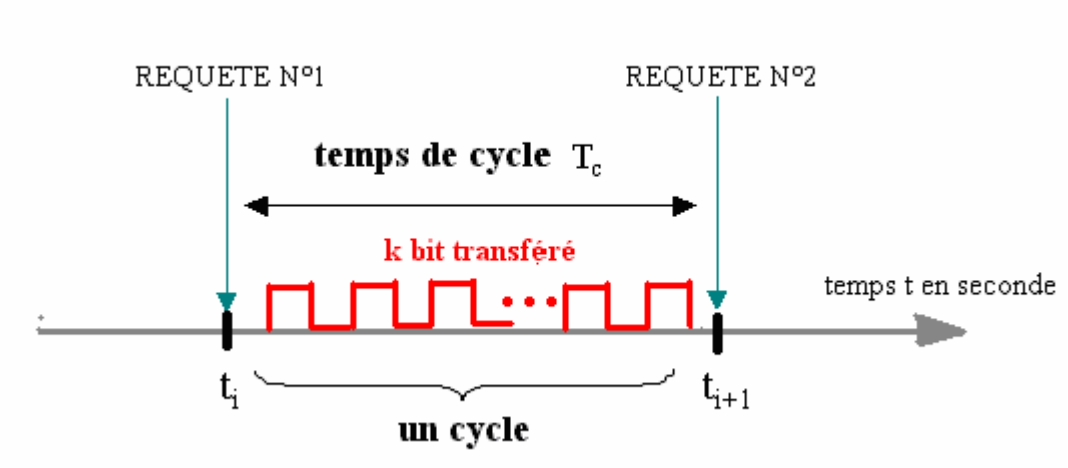


Figure I.4 Illustration d'un cycle

Il s'en suit que

$$F = \frac{1}{T_c}$$

F : fréquence en Hertz (Hz)

T_c : temps de cycle en seconde (s)

- **Débit d'une mémoire**

Globalement, le débit B exprime le nombre k de bits transférés au circuit extérieur par unité de temps pour d'autres (Tisserant 2003)

- **Volatilité de l'information**

La volatilité caractérise la permanence des données contenues dans une mémoire et son altération vis à vis de la présence ou non d'une source d'alimentation (Mercouroff 1990) .

I.4) Notion d'adresse

Pour bien distinguer une case mémoire d'une autre, une adresse unique lui est attribuée. Pour cela, les **n** lignes d'adresse sont reliées au 2^n lignes de la matrice par l'intermédiaire d'un circuit décodeur d'adresse, servant à sélectionner une case parmi les m existantes.

I.5) Opérations de base dans une mémoire

La majorité des mémoires utilisées dans les circuits logiques acceptent deux opérations de base (Dowsing 1987). Ce sont :

- les opérations de lecture
- les opérations d'écriture.

I.5.1) Opérations de lecture

Pour faire des requêtes de lecture, il faut utiliser trois signaux et bien respecter leur ordre d'injection respectif dans la mémoire (Figure I.5).

Premièrement, l'adresse de la case à lire est introduite sur le bus d'adresse.

Deuxièmement, la ligne de commande R/W est soumise à un signal de niveau logique 0.

Enfin, pour collecter les bits de données mémorisés dans la case, l'état de la ligne de sélection CS est mis à 0.

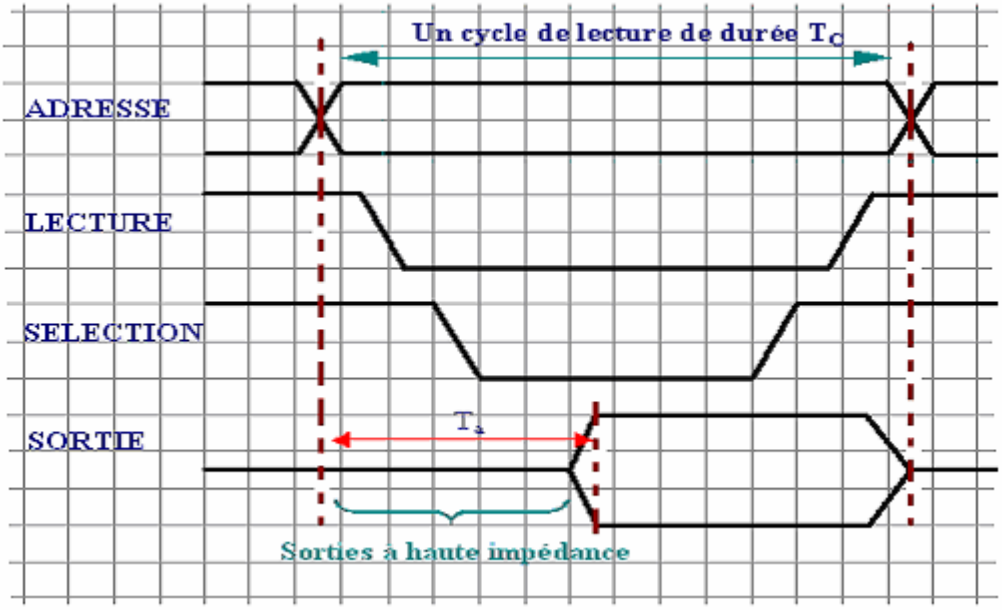


Figure I.5 Chronogramme d'un cycle de lecture

Remarques

1- Durant le temps d'accès T_a (la période entre la demande de lecture et l'obtention de l'information), les liaisons entre les sorties du bus de données et les cellules mémoires sont à haute impédance.

2- L'information recueillie au niveau d'une case reste présente sur le bus de données tout au long du cycle de lecture (Tisserant 2003).

I.5.2) Opérations d'écriture

Pour charger des données dans la mémoire, la procédure (figure I.6) consiste :

- tout d'abord, à établir l'adresse de la zone cible sur le bus d'adresse

- à sélectionner par la suite la case pointée par l'adresse en mettant à 0 le niveau de la ligne de sélection CS
- présenter les bits à enregistrer sur le bus de données
- à injecter une impulsion de niveau logique 1 sur la ligne de commande R/W pour valider la fonction d'écriture

Notons que l'adresse doit être stable avant de faire la sélection ; même chose pour les données avant d'appliquer l'impulsion d'écriture (Tisserant 2003).

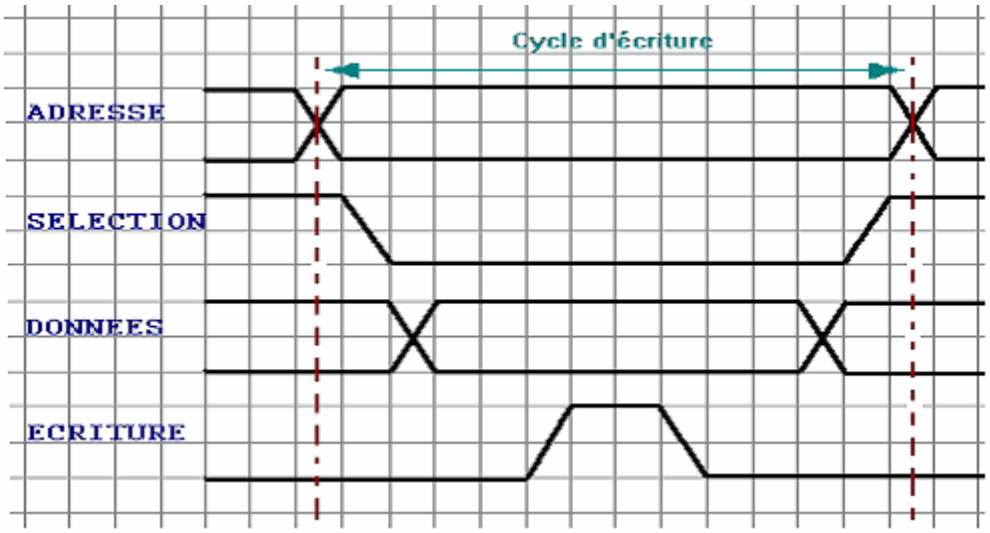


Figure I.6 Chronogramme d'un cycle d'écriture

I.6) Classification des Mémoires

Les mémoires sont classifiées suivant la volatilité des informations qu'elles contiennent. Elles sont principalement divisées en deux parties (Viennet 1998), ce sont les :

- mémoires vives
- mémoires mortes

I.6.1) Mémoires vives ou mémoires volatiles

Les mémoires vives, généralement appelées RAM, sont des mémoires dont la mémorisation ne s'effectue qu'en présence d'une source d'alimentation. Sur ces mémoires, l'accès aux informations est dit aléatoire puisque le temps d'accès T_a à une donnée est indépendant de sa position en mémoire (Hirsch 1992).

En effet, connaissant l'adresse d'une case, il est possible d'accéder directement à son contenu.

I.6.2) Mémoires mortes

A l'inverse des RAM, les mémoires mortes sont capables de conserver les informations qu'elles contiennent même si aucune alimentation électrique n'est disponible sur leurs bornes (Hirsch 1992).

Les mémoires mortes se divisent en deux grands groupes (Cattoen 2003)

- le premier groupe rassemble les mémoires à lecture uniquement ou ROM, conçu pour accomplir une tâche particulière définie par le constructeur pendant la fabrication en usine
- le deuxième groupe se rapporte aux mémoires programmables en une seule fois ou plusieurs fois. Avec cette variété de mémoire, il est possible de modifier leur contenus par programmation pour spécifier la mémoire à une tâche quelconque. Ce sont les :

  PROM : Programmable ROM

  EPROM : Erasable Programmable ROM

  EEPROM : Electrically Erasable Programmable ROM

I.7) La famille d'EPROM 27 à huit bits de données

Les EPROM 27 à 8 bits de données comme celui de la Figure I.7 sont des mémoires à accès aléatoire, effaçables avec un rayonnement ultra violet de longueur d'onde λ compris entre 230 et 253.7 nm.

Figure I.7 l'EPROM 27512 de Texas instruments

Le tableau I-2 suivant représente les caractéristiques de six EPROMs les plus courants.

Tableau I.2 Caractéristiques de quelques EPROMs

	EPROM 27xx					
CARACTERISTIQUES	**2716**	**2732**	**2764**	**27128**	**27256**	**27512**
CAPACITE (en kbit)	16	32	64	128	256	512
CAPACITE (en Ko)	2	4	8	16	32	64
NOMBRE DE CASE MEMOIRE	2048	4096	8192	16384	32768	65536
BUS DE DONNEES (en bit)	8	8	8	8	8	8

CHAPITRE II NORME RS 232 C et INTERFACAGE

II.1) Norme RS 232 C

Dans les systèmes numériques, la transmission de données peut se faire de deux façons :

- En parallèle : où tous les bits d'un mot de longueur quelconque sont transmis simultanément sur plusieurs fils.
- En série : où un caractère est transmis bit par bit sur un fil unique.

A cause du nombre élevé de fils qui constituent un bus parallèle, la transmission en parallèle convient mieux à des liaisons de courte distance (quelques mètres) et permet d'obtenir un débit numérique très important, tout en gardant sur chaque fil un débit en bit plus faible (Duplessix 1997).

Mais face à la nécessité de transmettre sur une longue distance, plusieurs types de transmission série ont vu le jour. Etant moins encombrante que la transmission en parallèle, elle utilise cette fois-ci la ligne téléphonique comme support de transmission pour relier des équipements distants de plusieurs kilomètres (Koren 1995).

Vers les années 60, l'EIA et le laboratoire Bell ont établi pour la première fois la standardisation de ces interfaces séries, afin que les équipements développés par les différentes firmes électroniques de cette époque puissent être connectés entre eux et sur n'importe quel terminal (Koren 1995). Ce standard est connu sous le terme de RS232 et se divise en trois normes qui sont : le RS232A, le RS232B et le RS232C ; mais parmi ces trois normes, c'est la norme RS232C qui est la plus utilisé (Sweet 1994). Notons qu'un standard similaire a été défini en Europe, par le CCITT sous la norme CCITT V.24/V.28.

II.1.1) Description d'une liaison RS232C

Sur une liaison RS232, les deux équipements reliés sont appelés DTE et DCE. DTE est le centre terminal de traitement de données (un ordinateur ou terminal) et DCE est le centre communicateur de données (Koren 1995). La connexion entre le terminal et le DCE est assurée par deux connecteurs mâle et femelle et un câble adéquat (Figure II-1).

D'après la norme, le connecteur mâle est placé du côté de DTE tandis que le connecteur femelle est placé du côté de DCE (Biggerstaff 1989).

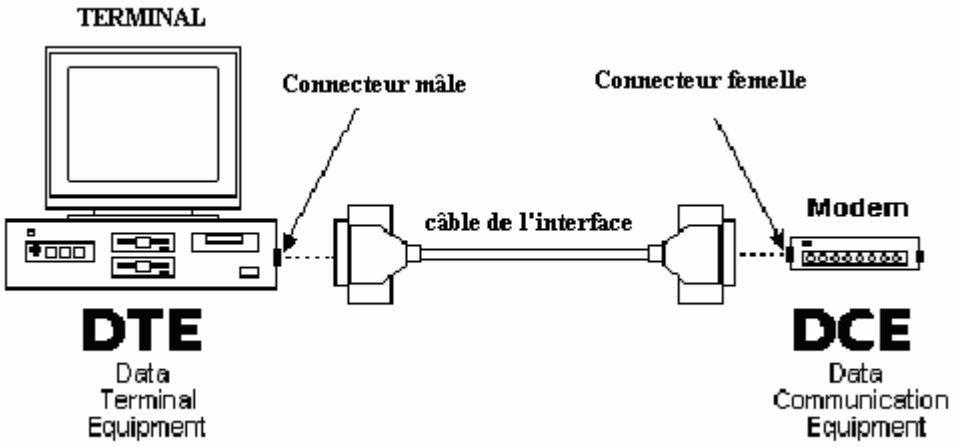


Figure II.1 Description d'une liaison RS232C

II.1.2) Signaux RS232C

Les signaux véhiculés sur les ports DB9 et DB25 peuvent être divisés en deux grands groupes suivant leurs rôles respectifs (Biggerstaff 1989) :

- Les signaux de données

Tableau II.1 Les signaux de données

SIGNAL	DIRECTION
R_x	DCE → DTE
T_x	DTE → DCE

- Les signaux de contrôle

Tableau II.2 Les signaux de contrôle

SIGNAL	DIRECTION
DTR	DTE → DCE
DSR/SDSR	DCE → DTE
RTS/SRTS	DTE → DCE
CTS/SCTS	DCE → DTE
DCD/SDCD	DCE → DTE
RI	DCE → DTE

II.1.3) Caractéristiques

- **Ports RS232C**

Sur la face arrière des ordinateurs, il existe deux variétés de port série RS232C :

- Le DB25 composé de 25 broches.

Figure II.2 Ports RS232C 25 broches

- le DB9 composé de 9 broches

Figure II.3 Ports RS232C 9 broches

- **Longueur du câble de transmission**

Au fur et à mesure que le signal progresse le long du câble de transmission, il s'atténue et finit par disparaître si le câble est trop long. Cet effet provient principalement de la capacité électrique du câble, dont la valeur fixée par le standard est de 2500 pF (Leibson 1986).

Par conséquent, pour avoir une transmission avec un niveau de bruit tolérable, la longueur nominale ne doit dépasser 17 mètres (Koren 1995).

- **Niveaux des signaux RS 232 C**

D'après Leibson (Leibson 1986), les signaux RS 232 C ne sont pas des signaux TTL (+5 V et 0 V) mais des signaux de niveau compris entre –15 V et +15 V.

Ainsi :

- sur les lignes de données T_x et R_x, le 1 logique (+5V en TTL) varie de –5 V à –15 V tandis que le 0 logique (0 V en TTL) appartient à l'intervalle [+5 V, +15 V] ;

- inversement, sur les lignes de contrôle, le bit «vrai», «on» ou « mark » (+5V en TTL) est compris entre +5V et +15V et le bit « faux », « off » ou « space » est défini entre -5V et –15V.

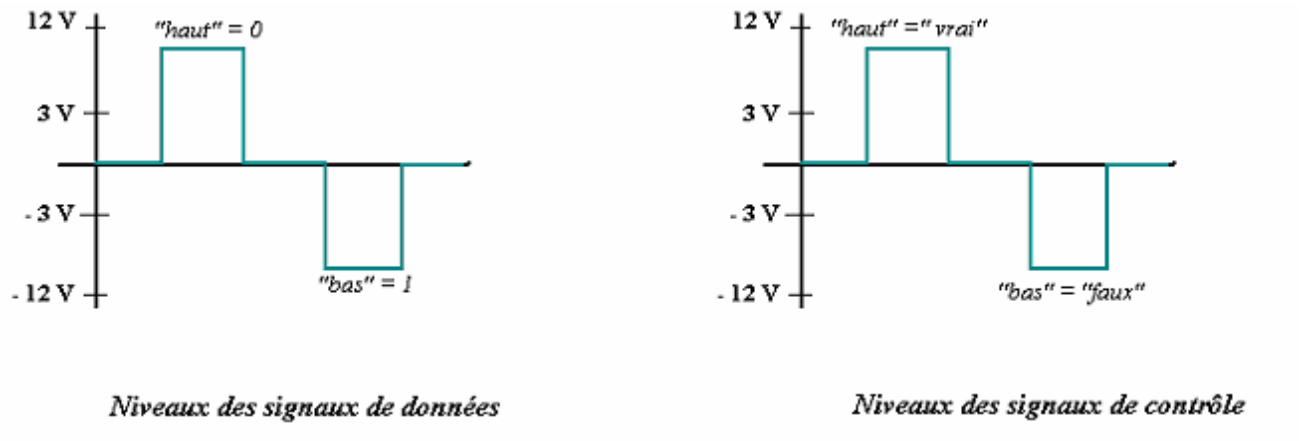


Figure II.4 Niveaux des signaux RS232C

- **Modes d'utilisation du bus RS232C**

Comme les autres voies de transmission, le bus RS232C (Margarotto 2003) peut être utilisé en mode :

- simplex

  le bus est utilisé dans une seule direction, l'un des deux équipements est l'émetteur pendant toute la session

- half duplex

  Le DTE et le DCE peuvent être successivement émetteur ou récepteur.

- full duplex

  Le bus est utilisé simultanément dans les deux sens, DTE et DCE sont émetteurs et récepteurs en même temps.

- **Débit d'une ligne RS232C**

Le débit, Baud rate ou rapidité de modulation (RM) d'une ligne RS232C est exprimé en Baud. où1 Baud est équivalent à un transfert de 1 bit par seconde (Hausmann 1988).

Si un bit est de durée δ (en seconde), alors le débit RM de la ligne s'écrit (Viennet 1998) :

$$RM=1/\delta$$

II.1.4) Communication série asynchrone

Transférer des données via le port RS232C sans aucune perte de bits exige une synchronisation stricte entre les deux équipements (DTE et DCE) pendant l'émission et la réception de données. La liaison série asynchrone est une technique qui peut être utilisée pour assurer cette synchronisation.

II.1.4.1) Principe

Pour ce type de liaison, le DTE et le DCE possèdent chacun une horloge interne qui fixe leur base de temps. La synchronisation est réalisée en encadrant les bits de données entre deux bits supplémentaires (Figure II-5), qui marquent le début et la fin du caractère émis tout en maintenant les horloges de DTE et de DCE à la même fréquence d'oscillation (Renesas 2003).

Le bit de début permet au receveur de synchroniser son horloge interne pour recevoir le caractère qui arrive. Après quoi, cette horloge est utilisée pour échantillonner le signal

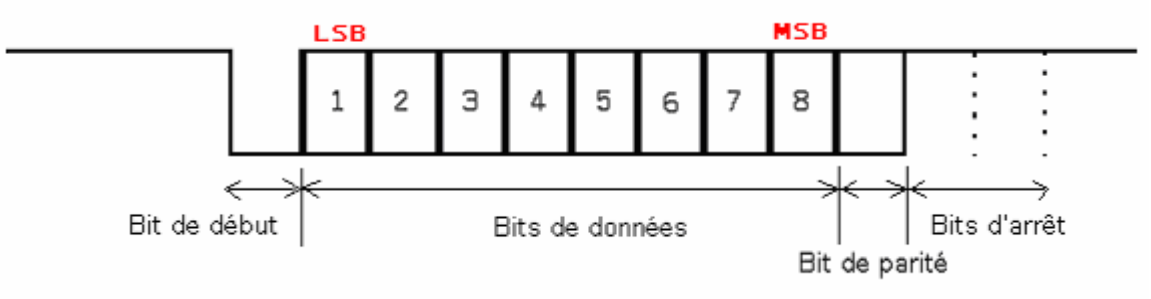


Figure II.5 Représentation d'une trame de caractère

II.1.4.2) Caractéristiques

- **Niveaux des bits de synchronisation**

Au repos, lorsqu'aucun caractère n'est transmis, les lignes de données T_x et R_x sont mis à « 1 ».

Ainsi, pour marquer le début et la fin d'une trame, le bit de début serait un bit de niveau bas et des bits de niveau haut pour le bit d'arrêt (Koren 1995).

- **Nombre de bits de données et de bits de synchronisation**

Une trame de données RS232C peut comporter :

- 8, 7, 6 , 5 ou 4 bits de données
- 1 seul bit de début

- 1, 1.5 ou 2 bits de fin

ces nombres dépendent du circuit qui effectue la sérialisation des données à transmettre, et du logiciel d'interfaçage.

- **Identification d'une trame asynchrone**

Dans la transmission asynchrone, une trame est symbolisée comme suit : « 8N2 », qui signifie que la trame est composée de 8 bits de données, sans bit de parité et 2 bits d'arrêt (Sweet 2003).

II.1.4.3) Détection d'erreur

Afin de vérifier la validité des bits de données reçus, un bit de parité est introduit par l'émetteur dans chaque trame ($9^{ème}$ bit de la Figure II-5). Pourtant, l'utilisation de ce bit n'est pas obligatoire puisqu'il est possible de l'activer ou de le désactiver pendant la configuration d'une session (Datasheet ISN8250).

Concernant les valeurs de ce bit (tableau II.3), ces valeurs dépendent :

- premièrement de la parité du nombre de bit de niveau 1 composant le caractère
- deuxièmement de la parité utilisée par les circuits d'acquisition des deux équipements.

Tableau II.3 Niveaux du bit de parité

PARITE UTILISEE	PARITE DU NOMBRE DE BIT « 1 »	BIT DE PARITE
Impaire (*Odd*)	Paire	1
	Impaire	0
Paire (*Even*)	Paire	0
	Impaire	1

Exemple : Supposons dans un premier temps que les deux équipements en liaison utilisent des signaux impairs. Pour envoyer la lettre C dont le code ASCII étendu est : 01000011, l'émetteur va insérer un bit de parité de niveau 0 dans la trame afin que le nombre de bit de niveau 1 reste impair.

Par contre, si la parité des deux équipements est paire, alors ce bit serait un bit 1.

Au cas où la détection du bit de parité aurait été désactivée (du type « None »), aucun bit ne serait inséré dans la trame

Remarques

1 Dans une liaison série asynchrone, les caractères peuvent être émis à tout moment
2 Les bits sont émis les uns après les autres, en commençant par le bit de poids faible

II.1.4.4) Poignée de main ou « handshaking »

La poignée de main est un procédé de reconnaissance fait par le terminal au niveau du DCE avant de transmettre des données (Biggerstaff 1989). Ce processus est utilisé afin d'interrompre momentanément la liaison si une interruption prioritaire se présente au niveau du microprocesseur du terminal.

Avant d'émettre des données (Figure II.6), le terminal met le signal DTR à 1 pour indiquer au DCE qu'il est prêt à établir la liaison, celui-ci manifeste sa disponibilité par l'émission d'un signal DSR=1.Une fois que le signal DSR est détecté par le terminal, un autre signal, RTS=1 est envoyé par le terminal pour demander au DCE l'autorisation de transmettre un caractère. De la même façon, DCE le répond par un signal CTS=1 s'il est prêt à recevoir le caractère.

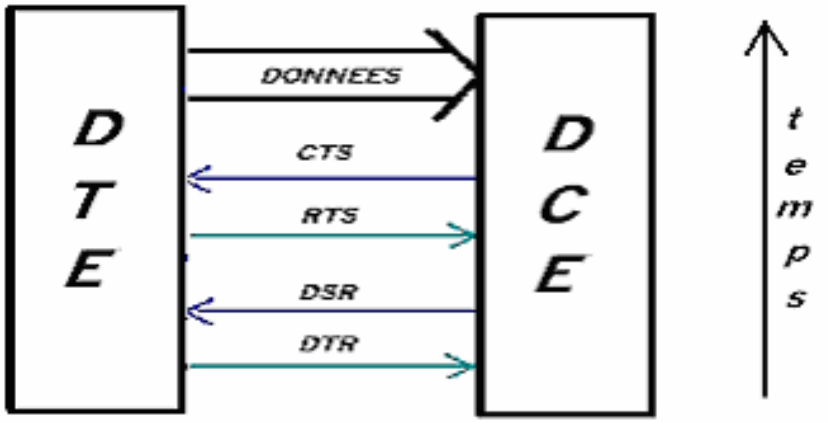


Figure II.6 Les protocoles de l'émission

Par contre (Figure II.7) ; pendant une réception de données, l'échange RTS/CTS n'est pas nécessaire puisque DCE peut envoyer directement des données aussi longtemps que le signal DTR soit au niveau 1.

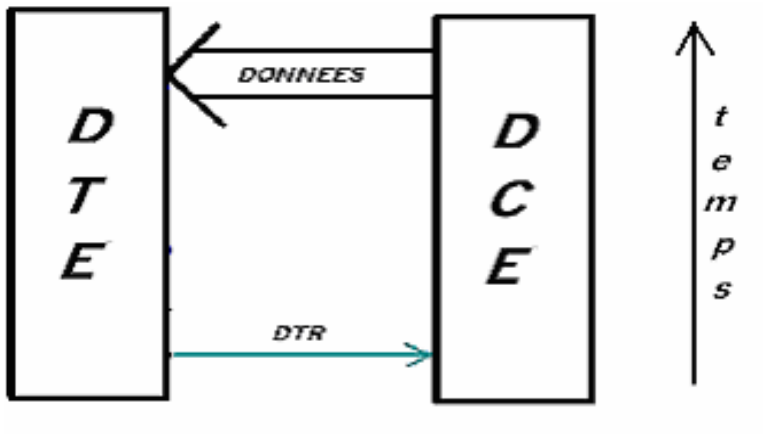


Figure II.7 Les protocoles de réception

II.1.5) Communication série synchrone

Pour ce type de liaison, deux bus sont utilisés (Margarotto 2003) :

- le premier est le support de transmission des flots de données synchrones
- le deuxième sert à transporter un signal d'horloge pour assurer la synchronisation des deux équipements pendant le transfert.

Pour marquer le début de la trame synchrone, un caractère spécial appelé drapeau, dont le code dépend du protocole utilisé est insérer au début et à la fin de la trame (Sweet 1994) et l'extraction des caractères qui composent la trame se fait à chaque front descendant d'horloge (Hirsch 1992).

II.1.6) Exemples d'interface série RS232C

Une liaison RS232C peut être utilisée entre un ordinateur et un autre équipement série comme une imprimante, un modem… ou entre deux ordinateurs en utilisant un câble null-modem.

II.1.6.1) Interface simple sans contrôle de flux

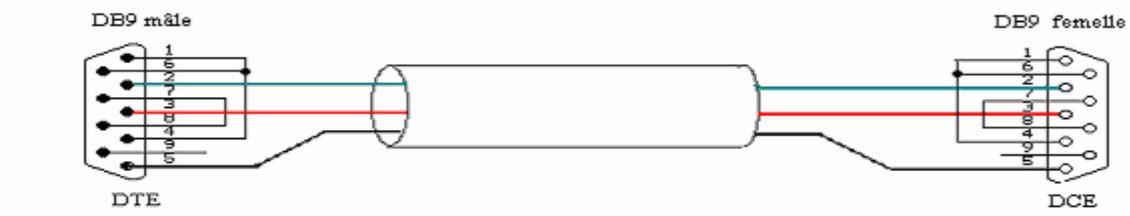


Figure II.8 Liaison Terminal-Périphérique sans contrôle de flux

II.1.6.2) Interface complète avec contrôle de flux

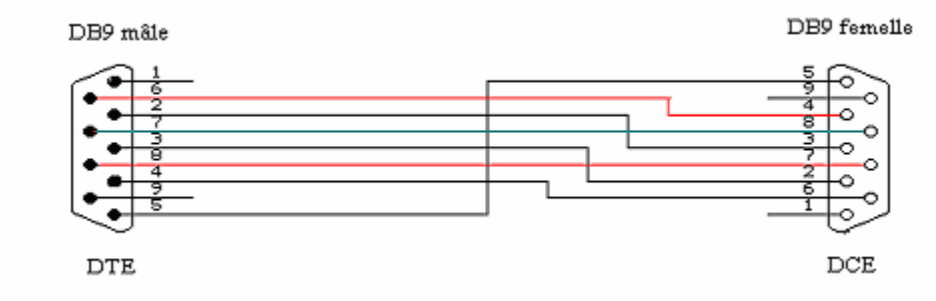


Figure II.9 Liaison Terminal-Périphérique avec contrôle de flux

II-1-6-3 Configuration d'un câble null-modem

Un câble null-modem est un câble utilisé pour relier deux ordinateurs distants de quelques dizaines de mètres, sans intercaler deux modems entre les deux en reliant le T_x de l'un au R_x de l'autre (Viennet 1998).

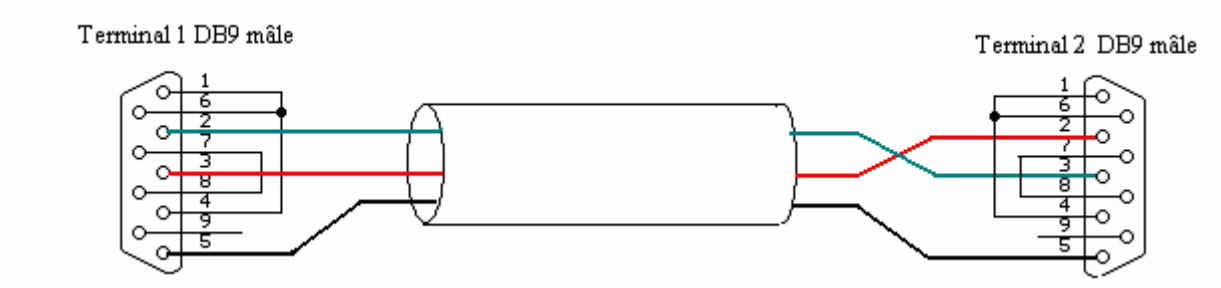


Figure II.10 Configuration d'un câble Null-Modem

II.2) Interfaçage

II.2.1) Définition

- Selon Larousse, une interface est l'ensemble des règles et des conventions qui permettent l'échange d'informations entre deux systèmes donnés.
- Matériellement, c'est le point de connexion qui assure l'échange de données entre ces deux systèmes.

Pour cela, pendant la réalisation d'une interface, il faut non seulement tenir compte des problèmes d'incompatibilités entre les appareils à lier, mais aussi, une connaissance générale de l'architecture du terminal est indispensable pour connaître le nombre d'interfaces nécessaires, responsables des traitements de données venant du DCE jusqu'aux résultats voulus.

II.2.2) Architecture de base d'un ordinateur

D'après Viennet (Viennet 1998), un ordinateur est une machine, capable d'acquérir de l'information, de la stocker, de la transformer en effectuant des traitements quelconques, puis de la restituer sous une autre forme.

D'après W.Mercouroff (Mercouroff 1990), c'est une machine composée de deux unités :

- l'unité centrale qui permet la mémorisation des programmes et des données
- l'unité d'échange qui permet les communications avec l'extérieur.

L'unité centrale se divise à son tour en deux parties (Huber 1998) :

- le processeur qui exécute les programmes et traite les données de la mémoire
- la mémoire principale, lieu de stockage de données et des programmes

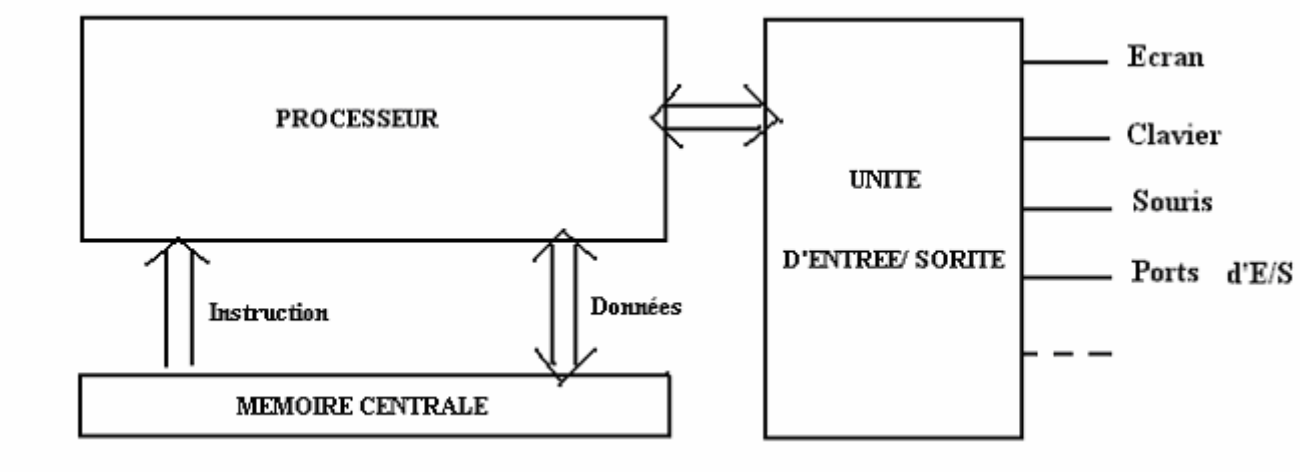


Figure II.11 Architecture de base d'un ordinateur

II.2.3) Compatibilité émetteur-récepteur

Rappelons qu'au niveau de l'ordinateur, l'information doit être représentée sous forme binaire (cf. chap1).

Or, la représentation des informations dans les divers systèmes informatiques n'est pas nécessairement la même selon les constructeurs et l'année de production de ces matériels.

Ainsi, la connexion de ces appareils exige l'existence d'un nouveau dispositif qui assure l'adaptation entre les données de l'émetteur et celles du récepteur (Hausmann 1988).

II.2.4) Interfaçage avec un micro-ordinateur

En effet, l'acquisition de données sur un terminal nécessite deux types d'interfaces :

- l'interface matérielle
- l'interface logicielle.

II.2.4.1) Interface matérielle

L'interface matérielle ou « interface block » est un système électronique câblé destiné à soulever les problèmes de compatibilité entre les équipements en liaison (Huber 1998) en tenant compte :

- des types de connecteurs utilisés
- des propriétés des signaux manipulés

II.2.4.2) Interface logicielle

L'interface logicielle est un système programmé logé dans la mémoire centrale, servant d'instruction au processeur pour traiter et stocker les données venant de l'interface matérielle. Cette partie assume le bon fonctionnement de l'ensemble du système depuis l'acquisition des données au niveau des circuits d'acquisitions jusqu'à la présentation et l'appréciation des résultats sur l'écran (interface graphique ou visuelle).

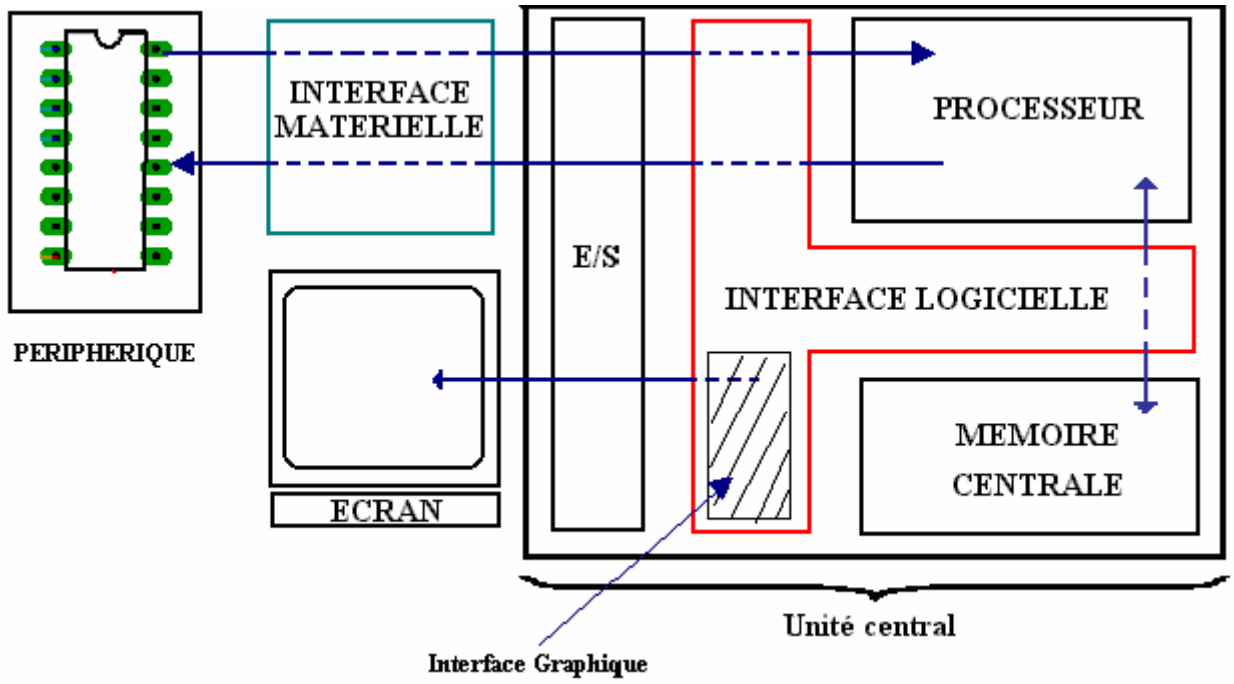


Figure II.12 Représentation des interfaces matérielle et logicielle

CHAPITRE III APPAREILLAGES et METHODOLOGIES

III.1) Appareillages

De la conception de l'interface jusqu'à sa réalisation, les matériels utilisés peuvent être divisés en deux parties :

- la première concerne le programmateur
- la dernière se rapporte aux outils logiciels utilisés.

III.1.1) Programmateur d'EPROM

Le programmateur utilisé est un programmateur spécialisé pour lire ou écrire sur un EPROM appartenant à la famille 27. Il se connecte sur l'interface par l'intermédiaire d'un connecteur de 25 broches du type DB25. et selon NJARASOATSINJOAVO Tsitohaina, le programmateur utilise une impulsion d'écriture de durée 5 ms pour écrire un caractère.

III.1.2) Outils logiciels

Pendant la conception et la réalisation de la partie électronique, les simulations sur ordinateur des circuits ont été faites avec le logiciel circuit maker Student version 6.2c de Protel. Tandis que la réalisation sur circuit imprimé à été effectuée en utilisant le logiciel de routage WinCircuit 2004.

Le logiciel d'acquisition de l'interface a été développé avec le logiciel C++ Builder version 5.0 de Borland. C'est un logiciel utilisant le langage C++ orienté objet et destiné à concevoir des applications interactifs multifenêtrages. Nous avons choisi ce logiciel à cause de la puissance de son compilateur et la possibilité d'obtenir un fichier exécutable après chaque compilation.

III.2) Méthodologies

III.2.1) Réalisation des circuits imprimés

Vu la complexité du traçage des circuits numériques manuellement, les cartes d'acquisition de l'interface ont été réalisées avec la méthode de photo-gravure à l'ultraviolet. Pour faciliter

la réalisation et la vérification des circuits imprimés après gravure, l'interface matérielle est divisée en deux cartes filles qui sont :

- la carte de configuration regroupant l'adaptateur de niveau et les circuits de configuration
- la carte de transmission qui assure les transferts de données

III.2.2) Type de transmission

Au niveau de la carte de transmission, le transfert de données entre le terminal et le programmateur utilise les concepts d'une liaison série asynchrone en mode semi-duplex. Ce transfert est assuré par le circuit UART 8250 de National Semiconductor.

Sachant qu'un UART est un circuit intégré universel émetteur et récepteur de signal RS232C asynchrone, réalisant automatiquement au rythme d'un signal d'horloge:

- la conversion d'une donnée parallèle de huit bits en signal asynchrone série
- la désérialisation d'un signal série asynchrone en donnée parallèle de huit bits.

Tout circuit intégré de type UART est pourvu au minimum de trois blocs de circuits (James 2003), qui sont :

- un bloc de transmission série
- un bloc de réception série (désérialiseur)
- un bloc d'interface pour la liaison avec un microprocesseur

Mais sur le 8250 (Leibson 1985) il y a en plus :

- un générateur de fréquence interne
- une ligne de commande de modem et d'état supplémentaire
- une structure d'interruption très complexe.

III.2.3) Programmation

Dans la partie programmation, le travail est fondé sur les concepts de la programmation orientée objet ; où la complexité du problème est subdivisée en plusieurs petits problèmes liés chacun à des événements extérieurs ou intérieurs attachés à l'interface. Cette subdivision facilite l'écriture des codes sources de l'interface et sa mise à jour en cas de besoin. Au cours de l'exécution du programme, son évolution dépend uniquement de l'état des événements associés à chaque module de programmes

CHAPITRE IV LA CARTE DE CONFIGURATION

Afin d'éviter des pertes d'informations, le terminal et l'interface doivent être en accord avant tout transfert de données. Ainsi, la configuration de l'interface est une tâche primordiale.

De plus, le 8250 est un circuit intégré qui travaille en dépendance d'un microprocesseur comme le 8086 de Intel, le Z80 de Zilog ou le 68000 de Motorola etc.… Ce dernier lui fournit les paramètres nécessaires à la communication tel que le nombre de bits de stop, débit de transfert, le type du bit de parité…

Or, l'inexistence d'un microprocesseur dans nos matériels à cause de la difficulté de trouver ce composant à Madagascar nous a obligé à concevoir une carte capable de diriger le 8250 et capable d'effectuer toutes les configurations nécessaires.

IV.1) Objectifs

La carte de configuration assure :

- le chargement des paramètres de la communication dans les registres du 8250 (bit de stop, bit de parité, débit,…)
- la commande des remises de l'interface à zéro en cas de besoin
- la direction du fonctionnement de l'UART ainsi que le programmateur en mode réception ou en mode transmission.

IV.2) Schéma synoptique

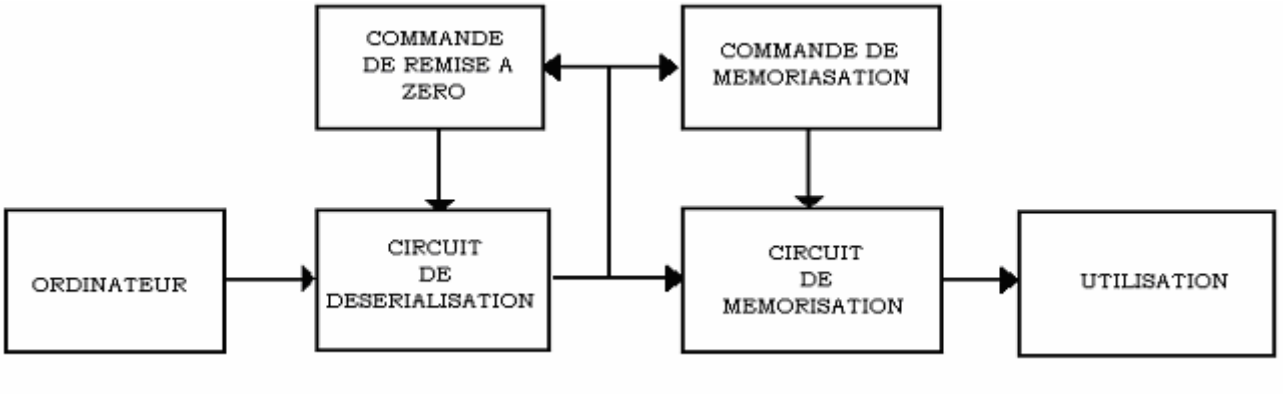


Figure IV.1 Schéma synoptique de la carte de configuration

IV.3) Fonctionnement

La carte de configuration est composée de quatre modules qui sont :

- le circuit de dé-sérialisation (désérialiseur)
- le circuit de mémorisation
- le circuit de commande de remise à zéro ou RAZ
- le circuit de commande de mémorisation

Sur son entrée, la carte reçoit une trame d'information de 24 bits série, obtenue en combinant les signaux RTS et DTR du port série.

IV.3.1) Signaux utilisés dans la carte

Sur la sortie du circuit de dé-sérialisation, les 24 bits parallèles obtenus peut être divisés en deux parties :

- la première est formée par les 20 premiers bits de faible poids, appelés signaux de configuration
- la deuxième est formée par les 4 bits restant appelés signaux de commande.

Avant d'être dirigé vers le circuit d'utilisation, les signaux de configuration passent tout d'abord dans le circuit de mémorisation. Avec ce procédé, les erreurs provenant d'un changement d'état non voulu au niveau de RTS ou de DTR n'aura pas d'influence sur l'information mémorisée dans le circuit de mémorisation. En outre, l'information reste présente à la sortie de la carte jusqu'à ce que la trame suivante arrive.

Par ailleurs, les signaux de commande sont dirigés vers le circuit de commande de mémorisation (MEM) et vers le circuit RAZ pour contrôler le fonctionnement de l'ensemble.

IV.3.2) Tables de vérité des circuits RAZ et MEM

Tableau IV.1 Table de vérité des fonctions RAZ et MEM

A_1	A_2	A_3	RAZ	MEM
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	1
1	1	0	1	0
1	1	1	0	0

Sur ce tableau, A_1, A_2 et A_3 sont respectivement le 21ème, 23ème et 24ème bit de la trame c'est à dire les bits du signaux de commande.

IV.4) Commande de remise à zéro

D'après cette table de vérité, la fonction RAZ s'écrit :

$$RAZ = A_1 * A_2 * \overline{A_3}$$

A l'aide d'une porte AND à trois entrées et d'un inverseur, ce circuit se présente comme suit

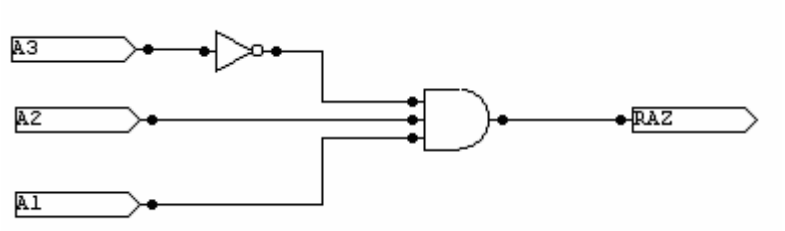


Figure IV.2 Schéma du circuit RAZ

IV.5) Désérialiseur

Le désérialiseur est l'élément de base de la carte ; c'est un circuit à deux entrées et 24 sorties, basé sur la transformation série parallèle effectuée par trois registres à décalage de 8 bits, le SN74164

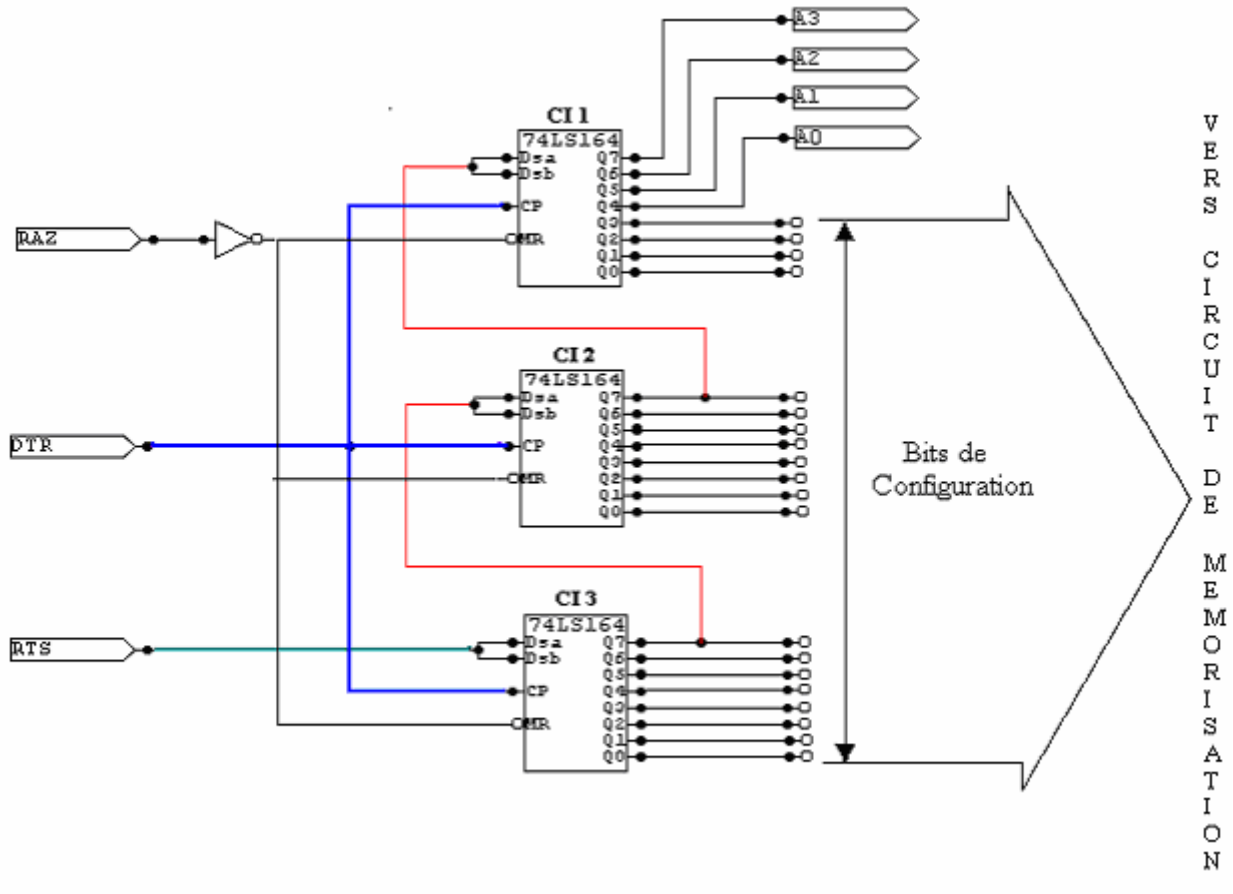


Figure IV.3 Schéma du circuit désérialiseur

IV.5.1) Fonctionnement

Tout d'abord, notons que le signal RTS du port série est le support qui transporte la trame de 24 bits énoncée plus haut. Tandis que DTR constitue le signal d'horloge qui cadence l'injection de la trame dans les registres à décalage.

Sur ce schéma, les trois registres CI 1, CI 2 et CI 3 sont mis en cascade pour assurer la progression des bits, de CI 3 jusqu'à CI 1 lorsque les capacités de CI 3 et de CI 2 sont dépassées.

En effet, lorsque plus de 8 bits sont introduits dans CI 3, les bits qui apparaissent sur la sortie Q7 sont immédiatement injectés dans le deuxième registre par l'intermédiaire de ses deux entrées dsa et dsb. De cette manière, les excès de bits au niveau de CI 3 peuvent être récupérés sur les sorties Q0 à Q7 de CI 2. De la même façon, pour récupérer les excès de bits

au niveau de CI 2 lorsque plus de 16 bits sont introduits dans CI 3, sa sortie Q7 est connectée sur les entrées dsa et dsb de CI 1.

La remise à zéro de l'ensemble est assurée par le circuit RAZ décrit précédemment. Seulement, à cause de l'état d'activation des pins $\overline{MR}$ des registres à l'état 0, le signal RAZ est tout d'abord inversé avant d'être connecté sur ces pins.

En résumé, ce circuit est identique à un registre à décalage de 24 bits à entrée série et sortie parallèle.

IV.6) Circuit de mémorisation

La mémorisation des 20 bits de configuration est réalisée par trois circuits intégrés du type octal D à horloge commune, le SN74273.

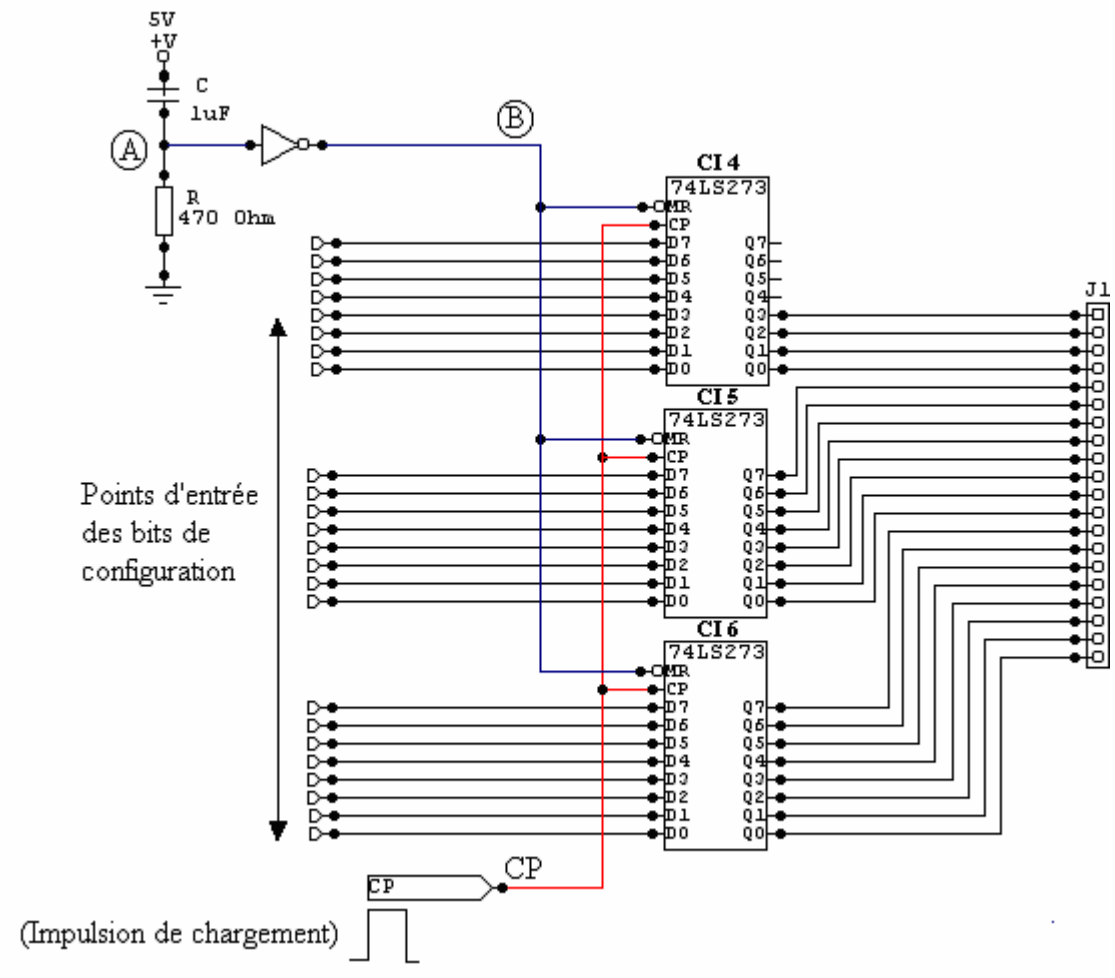


Figure IV.4 Schéma du circuit de mémorisation

IV.6.1) Fonctionnement

Les 20 bits de configuration venant du désérialiseur sont injectés sur les entrées D0 à D7 de CI 6 et de CI 5 et sur les entrées D0 à D3 de CI 4. La mémorisation de ces bits est effectuée en appliquant une impulsion provenant du circuit de commande de mémorisation sur les pins CP des trois circuits intégrés.

Au cours de la réalisation, nous avons constaté que les sorties des circuits intégrés SN74273 présentent des états indéterminés juste après la mise sous tension de la carte. Pour cela, une remise à zéro des trois circuits intégrés doit être exécutée après chaque alimentation, le circuit responsable de ce rejet est un circuit RC classique constitué par un condensateur de capacité 1µF et d'une résistance de 470 Ω. Ce circuit génère une impulsion de durée t_1 = 0.47 ms (t = R.C) au nœud A et un signal de durée t_2 = 0.33 ms (t_2 = RCln2) sur le nœud B.

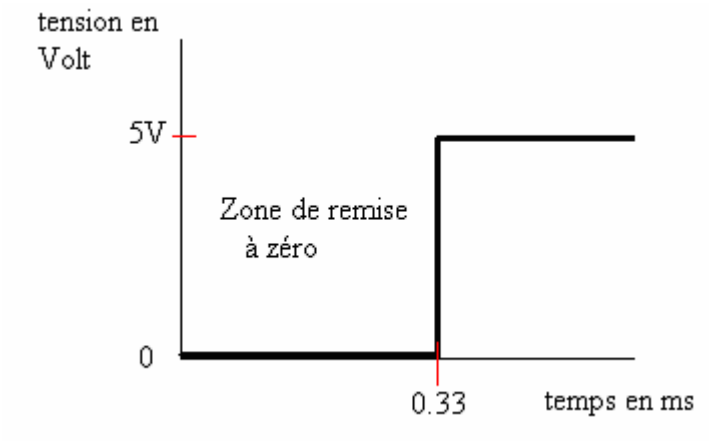


Figure IV.5 Chronogramme obtenu en B

IV.7) Circuit de commande de mémorisation

IV.7.1) Description

D'après la table de vérité précédente, la fonction qui définit la commande de mémorisation MEM s'écrit

$$MEM = A_1 * \overline{A_2} * A_3$$

Comme le circuit RAZ, à l'aide d'une porte AND à trois entrées et un inverseur, ce circuit se présente comme suit

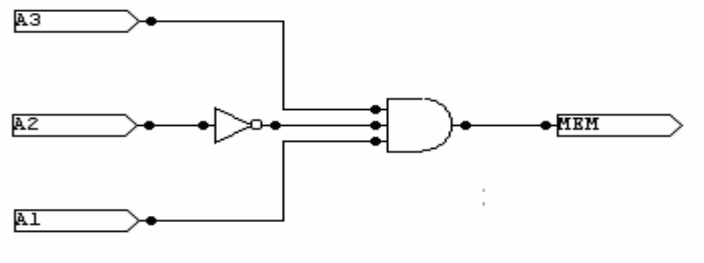


Figure IV.6 Schéma du circuit MEM

Pendant le premier test, la sortie du circuit MEM a été directement reliée sur la ligne CP des bascules du circuit de mémorisation. La mémorisation des bits de configuration s'effectuait parfaitement ; seulement, des erreurs ont été notées sur les sorties de ces bascules.

Ces erreurs ont été dues :

- premièrement à l'instabilité des sorties des registres CI 3 et CI 2 lorsque la fonction MEM est vérifiée. Au moment où le circuit MEM délivre le signal de commande, la transformation série-parallèle de la trame effectuée par ces registres n'est pas encore complètement terminée ;
- deuxièmement, nous avons remarqué que les trois octal D du circuit de mémorisation continuaient à charger les bits qui se trouvent sur ses entrées tant que le signal injecté sur CP reste à 1. Un changement d'état non voulu au niveau des sorties des registres parviendrait donc à modifier facilement la configuration mémorisée précédemment.

Pour résoudre ce problème, le signal délivré par le circuit MEM est tout d'abord acheminé dans une ligne à retard afin d'attendre que les sorties des registres soient stables, puis injecté dans un monostable à porte NAND pour fixer la durée de chargement des bits dans le circuit de mémorisation (Figure IV.7).

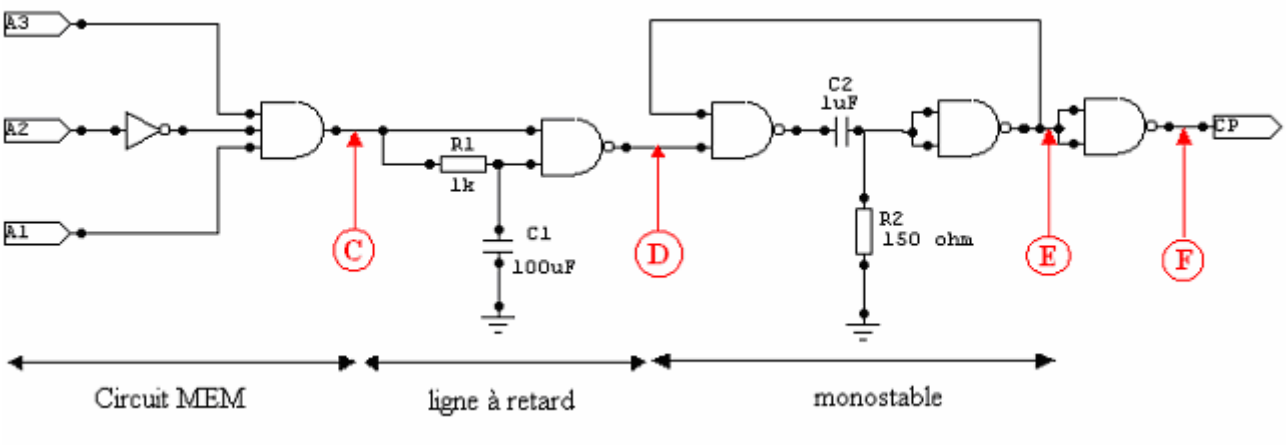


Figure IV.7 Schéma du circuit de commande de mémorisation

IV-7-2) Chronogrammes obtenus sur C, D, E et F

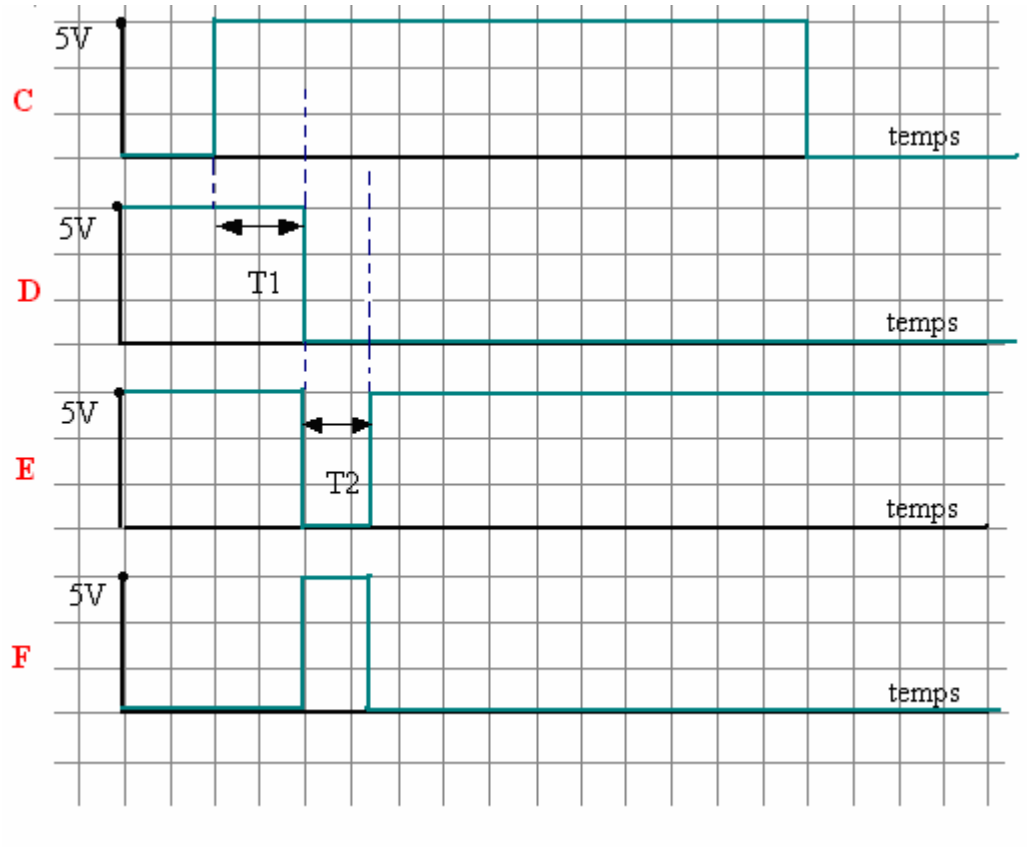


Figure IV.8 Chronogrammes obtenus aux nœuds C, D, E et F

Sur ces chronogrammes,

$$T1 = R1 * C1 * \ln 2 = 0.69 \text{ ms} \qquad \text{et} \qquad T2 = R2 * C2 * \ln 2 = 0.10 \text{ ms}$$

Remarque :

Contrairement aux signaux TTL manipulés par la carte de configuration, les signaux RTS et DTR sont des signaux RS 232. Par conséquent, un MAX232 est placé juste avant le désérialiseur pour les adaptés au niveau TTL.

IV-8) Schéma définitif de la carte de configuration

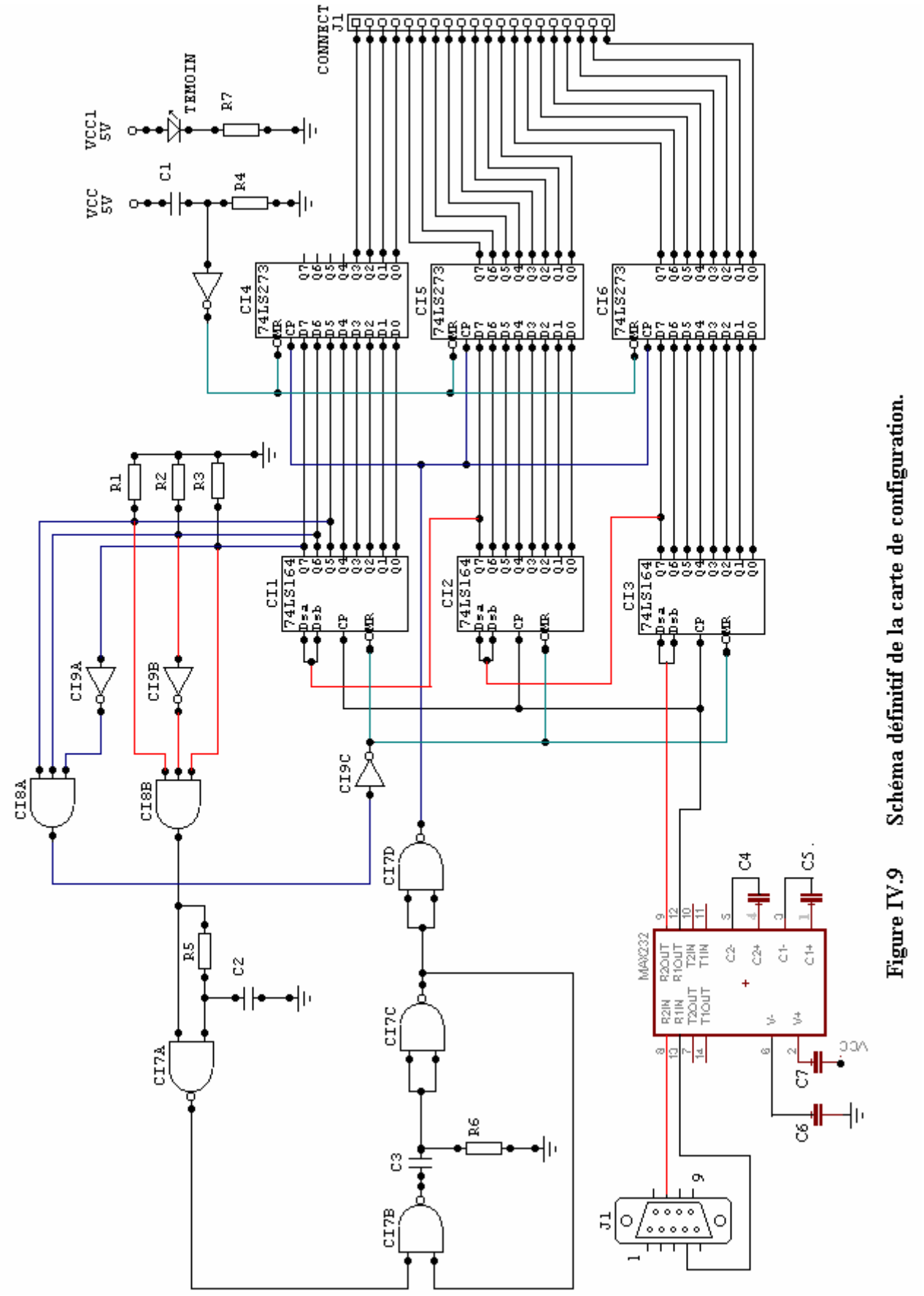


Figure IV.9 Schéma définitif de la carte de configuration.

CHAPITRE V LA CARTE DE TRANSMISSION

V-1) Schéma de principe

La carte de transmission est le circuit qui sert d'interprète entre le programmateur et le terminal.

C'est le circuit essentiel de l'interface matérielle, reposant sur le schéma de principe suivant.

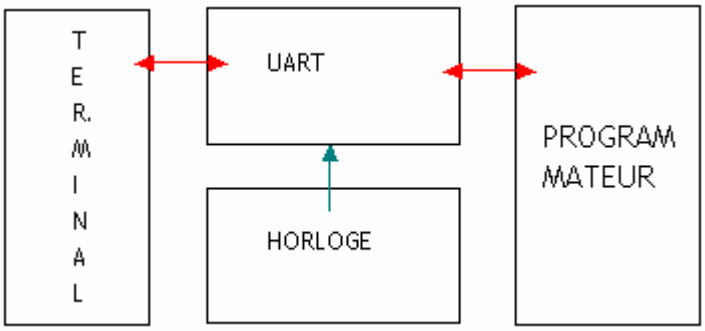


Figure V.1 Schéma de principe de la carte de transmission

V.2) Fonctionnement

Le circuit présenté ci-dessus est composé de deux unités et assure la liaison terminal-programmateur en mode semi-duplex uniquement.

En mode écriture, les données venant du terminal sont désérialisées par l'UART au rythme du signal l'horloge avant d'atteindre le programmateur. Réciproquement, en mode lecture, elles sont sérialisées par la même unité avant d'accéder au port série.

Mais à cause de la différence de niveau entre les signaux RS232 du terminal et ceux manipulés par la carte, un adaptateur de niveau à base de MAX 232 est intercalé entre ces deux unités. Cet adaptateur est placé du côté de la carte de configuration puisque la carte de transmission demande beaucoup plus d'espace. Par ailleurs, ce circuit est utilisé par les deux cartes.

V.3) Présentation de chaque unité

V.3.1) Adaptateur de niveau

L'adaptateur de niveau de la figure suivante utilise un MAX 232 et 4 condensateurs de 1µF pour transformer les signaux RS232 en signaux TTL et inversement.

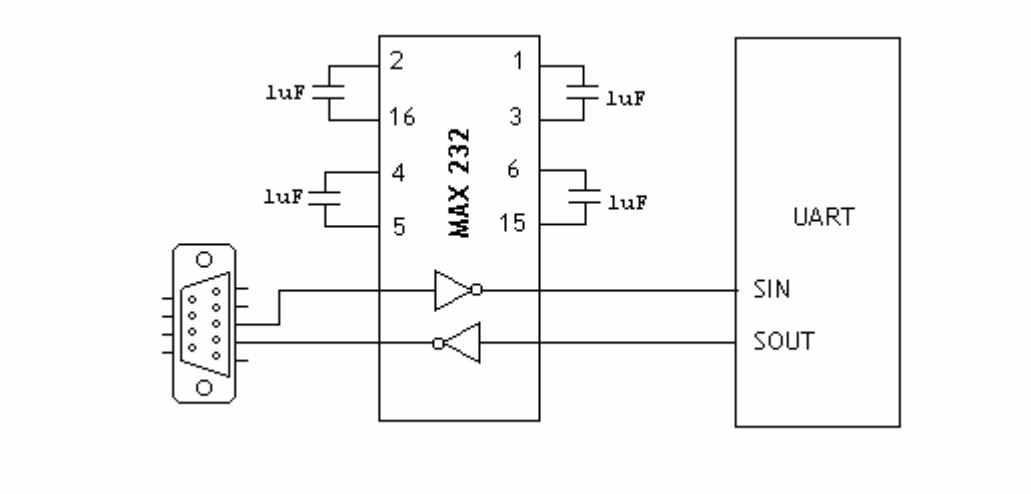


Figure V.2 Schéma de l'adaptateur de niveau

Ce montage fonctionne sous 5V ; les quatre condensateurs de 1µF constitue avec les circuits interne du Max les circuits de pompage responsable de la conversion de la tension de 5V en +10V et –10V (Maxim 1996).

V.3.2) UART

Il est déjà dit précédemment que l'UART utilisé est le 8250 de National Semiconductor. Toutefois, vu la ressemblance entre l'architecture de base d'un 8250 et des UARTs comme : le 16450, 16550, 16650, … , il est également possible d'utiliser l'un de ces UARTs, sans changer le montage.

V.3.3) Sélecteur de bus

V.3.3.1) Problème rencontré

La particularité du 8250 par rapport aux autres UARTs comme l'AY-5-1013 de General Instrument ou le CDP 6402 de Harris Semiconductor, qui le rend difficile à manipuler est la présence d'un bus de données et d'un bus de contrôle confondus sur un même bus bidirectionnel.

V.3.3.2) Solution adoptée

Pour cela, afin de bien distinguer les signaux qui vont entrer dans le 8250 et ceux qui vont en sortir, nous avons connecté sur le bus de données du 8250 trois circuits intégrés transmetteurs de bus bidirectionnel, le SN74245 (figure V.3).

Le premier transmetteur nommé TRANS1 se charge du transfert des bits de configuration et est polarisé dans le sens ordinateur-UART.

Le second transmetteur nommé TRANS2 s'occupe de la vérification des paramètres inscrits dans les registres de l'UART et polarisé dans le sens UART-Circuit de visualisation composé de huit LEDs.

Concernant le troisième transmetteur (TRANS3), il se charge des transferts de données entre l'UART et le programmateur ; sa polarisation dépend de la nature de l'opération à effectuer (lecture ou écriture).

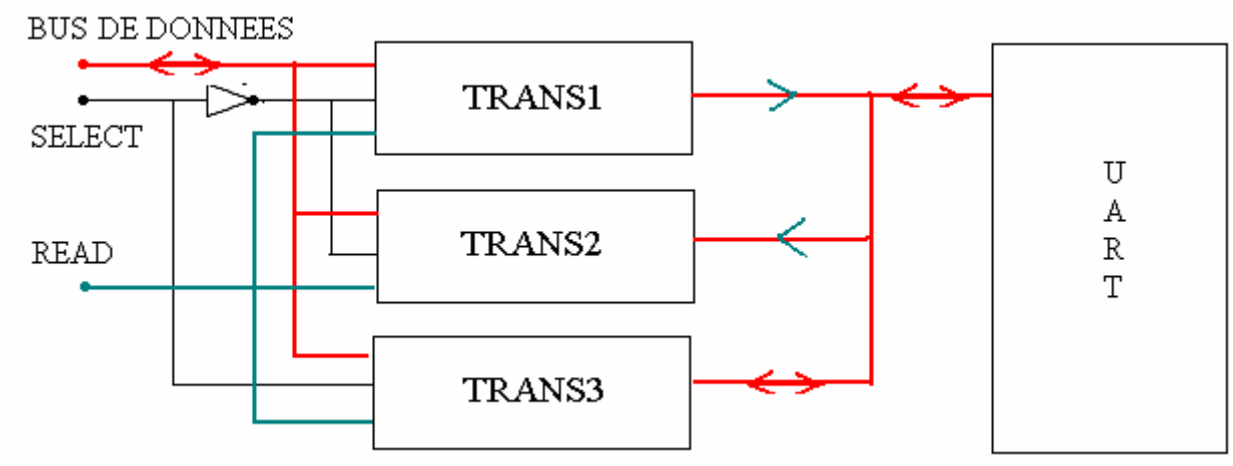


Figure V.3 Schéma des sélecteurs de bus

V.3.3.3) Principe de fonctionnement

Pendant la configuration de l'UART et du port de communication (COM), un seul transmetteur entre TRANS1 et TRANS2 peut être utilisé. Ce choix dépend des niveaux des signaux SELECT et READ présentés dans le tableau suivant.

Tableau V.1 Rôles de TRANS1 et TRANS2

SELECT	READ	Transmetteur	Tâche
1	0	TRANS1	Ecriture des bits de configuration dans l'UART
1	1	TRANS2	Lecture des paramètres de configuration de l'UART
0	X	X	X

Par contre, en mode transmission de données, seul TRANS3 fonctionne. Notons que le basculement entre le mode configuration et le mode transmission de données dépend du niveau du signal SELECT ; si ce signal est mis à zéro alors c'est le mode transmission qui est active.

Comme précédemment, le tableau suivant présente l'état de TRANS3 vis à vis des niveaux des signaux SELECT et READ.

Tableau V.2 Polarisations et rôles de TRANS3

SELECT	READ	Polarisation de TRANS3	Tâche
0	0	Programmateur-UART	Ecriture des bits de données dans l'UART
0	1	UART-Programmateur	Envoi des bits de données vers le programmateur
1	X	X	X

Remarques

1) Lorsque le composant SN74245 n'est pas polarisé, il se trouve dans un troisième état dit haute impédance. Au cours des essais, nous avons constaté que pendant un transfert de donnée, la transition entre l'état haute impédance et l'état passant de TRANS3 entraîne la modification des caractères envoyés ou reçus par l'UART. De ce fait, les huit lignes du bus de données de l'UART ont été polarisées vers la masse par l'intermédiaire de huit résistances de 3,9 kΩ.
2) Le SN74245 est activé en mettant un signal de niveau zéro sur la 19ème broche. C'est pourquoi, un inverseur est placé sur cette broche pour inverser le signal SELECT.

V.3.4) Horloge

Intérieurement, le 8250 est muni d'un oscillateur et d'un diviseur de fréquence, dont l'ensemble est appelé « baud rate generator » qui lui permet de générer la plupart des baud rates nécessaire à la transmission.

Pour activer ce générateur, deux solutions sont possibles :

- soit un quartz est connecté sur les pins 16 et 17 du 8250
- soit la sortie d'une horloge externe est connectée sur le pin 16 de l'UART

Dans notre cas, nous avons choisi la deuxième solution puisqu'il y avait des fois où le générateur interne de 8250 ne fonctionnait pas correctement avec un quartz externe.

Cet oscillateur externe (figure V.4) est constitué d'un compteur binaire (le CD 4060) et d'un quartz de 1.8432 MHz

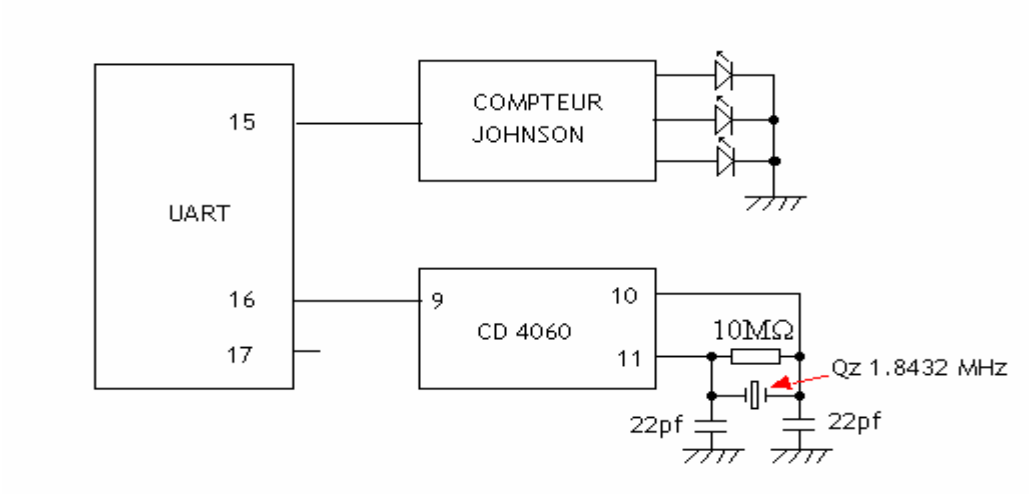


Figure V.4 Schéma du circuit d'horloge

V.3.4.1) Fonctionnement

Sur la borne 9 du compteur est disponible un signal sinusoïdale de fréquence égale à celle du quartz. Au moment où ce signal est introduit sur la 16ème broche de l'UART, un autre signal proportionnel au nombre introduit dans les registres du diviseur de fréquence apparaît sur le pin 15. Ce dernier est ensuite conduit vers un compteur Johnson afin de visualiser sur trois LED le bon fonctionnement de l'horloge.

V.4) Protocoles contrôleur de flux utilisés

Sur cette carte, les protocoles utilisés en mode écriture et en mode lecture sont différents.

V.4.1) Protocoles d'écriture

En mode écriture, c'est à dire l'envoi de données de l'ordinateur vers l'EPROM, ce sont les signaux RTS et CTS qui sont employés. D'après les caractéristiques du programmateur, une impulsion de 5 ms est nécessaire pour écrire un caractère de huit bits dans un EPROM. En conséquence, le CTS du port est mis à zéro au moment où le programmateur écrit le caractère ; l'ordinateur ne doit donc transmettre un nouveau caractère que si CTS est remise à un.

V.4.2) Protocoles de lecture

Pendant le transfert des données du programmateur vers l'ordinateur, trois signaux sont utilisés :

- le RTS du port série
- le signal INTRPT (pin 30) du 8250
- le signal LOAD venant de la carte de configuration

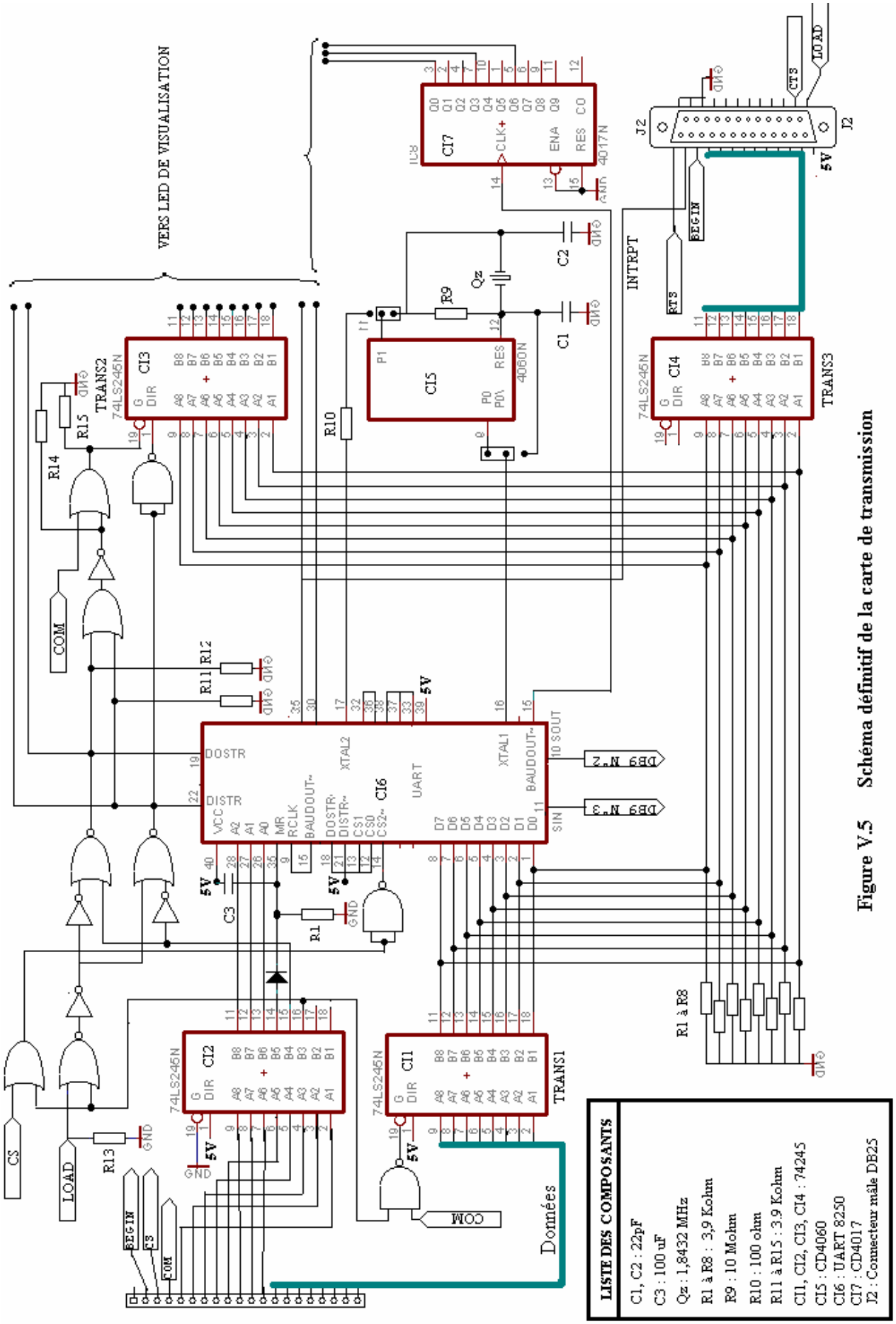
RTS et INTRPT servent à avertir le programmateur de l'état du terminal et de l'UART à prendre en compte le prochain caractère (tableau V.3) ; alors que LOAD est un signal envoyé par la carte de configuration pour amorcer ou arrêter le transfert de données par programme.

Tableau V.3 Signification des niveaux des signaux RTS et INTRPT

Signal	Niveau	Etat
RTS	0	Terminal occupé
	1	Terminal prêt à communiquer
INTRPT	0	UART occupé
	1	UART prêt à communiquer

Ces trois signaux sont envoyés vers le programmateur de sorte que si l'un d'entre eux est inhibé (niveau 0), le transfert de données s'arrête momentanément ou définitivement jusqu'à ce que les trois signaux redeviennent tous 1.

V.5) Schéma définitif de la carte de transmission



LISTE DES COMPOSANTS

C1, C2 : 22pF
C3 : 100 uF
Qz : 1,8432 MHz
R1 à R8 : 3,9 Kohm
R9 : 10 Mohm
R10 : 100 ohm
R11 à R15 : 3.9 Kohm
CI1, CI2, CI3, CI4 : 74245
CI5 : CD4060
CI6 : UART 8250
CI7 : CD4017
J2 : Connecteur mâle DB25

Figure V.5 Schéma définitif de la carte de transmission

CHAPITRE VI LE LOGICIEL D'ACQUISITION DE DONNEES

La réalisation de l'interface logicielle est une partie aussi importante que les parties électroniques décrites précédemment.

Par rapport à la carte d'interfaçage, elle se tourne plutôt vers la partie « soft ». Des programmes y sont donc écrits afin de diriger, de gérer et de commander le processus de transfert de données entre le programmateur et le terminal depuis le terminal lui-même. Sans eux, les circuits électroniques de la carte d'interfaçage sont inactifs.

VI.1) Objectifs

Les objectifs fixés dans la conception de l'interface logicielle sont :

- la définition des fonctions qui assure :
    - les configurations de l'UART
    - l'acquisition et l'échange de données au niveau du port série de l'ordinateur
- la réalisation d'un éditeur hexadécimal et ascii pour la présentation et la collection de données
- la réalisation d'un gestionnaire de fichiers et d'archivage, objectif principal de ce travail
- la réalisation d'une interface graphique interactive facile à manipuler.

VI.2) Module de programme de pilotage du port série

Cette partie du programme est le noyau de l'interface logicielle. Elle prend en charge les différentes configurations à effectuer pendant le mode configuration et gère l'entrée et sortie de caractère au niveau du port pendant un transfert de données.

Pour faciliter le développement de l'interface, nous avons choisi de regrouper toutes les fonctions qui constituent ce module de programme dans un fichier DLL nommé SerialComm.dll

Avec ce procédé, il est possible de :

- réutiliser le module de gestion du port sans retoucher et recompiler ses codes sources.
- mettre à jour l'interface en ajoutant les nouvelles fonctions dans le DLL en cas de besoin sans toucher à l'interface graphique.

VI.2.1) Ouverture et configuration du port COM

Toute manipulation faite au niveau du port série, que ce soit en mode configuration ou en mode transmission, est bloquée tant que ce dernier n'est pas ouvert.

En langage C ou C++, la manipulation d'un port COM et la manipulation d'un fichier (texte ou binaire) sont deux choses similaires. En d'autre terme, à chaque port COM est associé un fichier du même nom.

En conséquence, l'accès port voulu se traduit par l'ouverture du fichier correspondant en mode écriture-lecture pour autoriser l'utilisateur à lire et à écrire sur le port.

Concernant les configurations du port, le chargement des paramètres tels que baudrate, bits de stop, bits de données et parité dans son contrôleur est réalisé avec les structures DCB suivantes

dcb.BaudRate

dcb.fParity

dcb.ByteSize

dcb.StopBits

Au niveau du DLL, ces deux opérations sont englobées dans la fonction OpenPort(COMId, Param) où

- COMId désigne l'identité du port (COM1 ou COM2) à ouvrir
- Param représente les paramètres de configurations à charger

L'appel à cette fonction permet donc à la fois de configurer le port COM voulu et de l'ouvrir.

VI.2.2) Configuration de l'UART

Rappelons qu'au niveau de la carte de configuration, 24 signaux parallèles sont obtenus grâce à l'introduction d'une trame de données série (appelée trame de configuration) par l'intermédiaire de DTR et de RTS sur son entrée. Sur ces 24 signaux, les 20 bits de faible poids constituent l'information à envoyer vers l'UART.

Par conséquent, à une trame de données (codée en hexadécimal) reçue par la carte correspond une information de configuration.

VI.2.2.1) Information de configuration

L'information de configuration est composée de deux parties :

- le paramètre à charger dans le registre de l'UART (du bit D_0 à D_7)

- l'entête (du bit D_8 à D_{19}), la partie qui indique l'adresse du registre cible et les opérations à faire sur ce registre.

VI.2.2.2) Organigramme du programme de configuration

Intérieurement, le 8250 est muni de 9 registres pour administrer la communication. Sur ces 9 registres, 5 seulement sont utilisés dans ce travail, ce sont :

- les deux registres du diviseur de fréquence DLL et DLM
- le registre contrôleur de ligne LCR
- le registre contrôleur de l'état du Modem MCR
- le registre d'interruption IER.

Le rôle du programme de configuration consiste donc :

- à calculer dans un premier temps le code hexadécimal qui associe l'entête (l'adresse du registre) et le paramètre de configuration
- a envoyer par la suite le code obtenu vers la carte de configuration sous forme de signal série de 24 bits en utilisant la fonction ConfigUART du DLL.

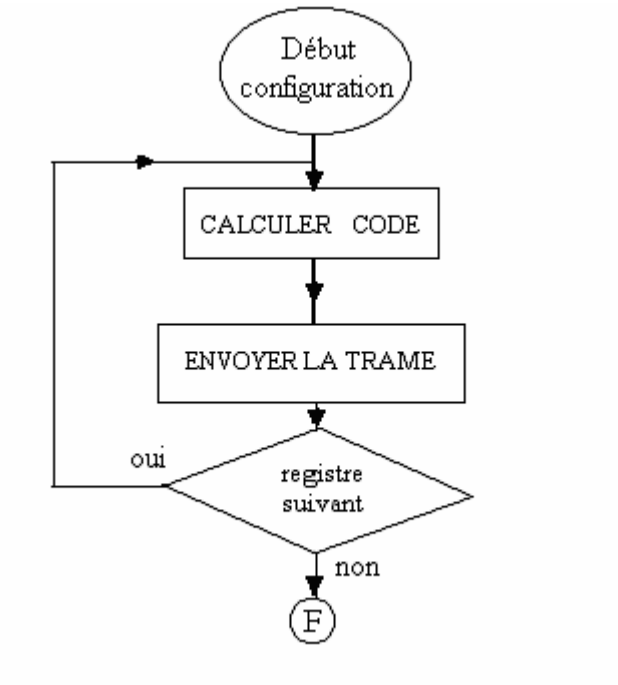


Figure VI.1 Organigramme du programme de configuration

Les différents codes hexadécimaux utilisés dans ce programme ainsi que les cinq registres parlés plus haut sont détaillés dans l'annexe II .

Remarque

Le signal émis par la fonction ConfigUART est obtenu en faisant basculer l'état de DTR et de RTS dans les bons niveaux au bon moment. Cette fonction prend comme paramètre le nombre (codé en héxadécimal) à envoyer vers la carte de configuration et retourne la valeur zéro dans le cas où elle échoue.

VI.2.3) Lecture de données

Tout d'abord, pour amorcer la lecture, le programme fait appel à la fonction ConfigUART du DLL. Cette fois ci le signal d'amorçage est envoyé vers le programmateur

Pour recenser les caractères qui arrivent sur le port, deux techniques peuvent être utilisées :

- soit on lance un « thread », un programme qui s'exécute en parallèle au programme principal et ne servant qu'à détecter l'arrivée d'un ou plusieurs caractères
- soit on lit chaque caractère à partir de l'événement qu'il produit à son arrivée.

Dans notre cas, c'est la première solution qui est adoptée en suivant le principe ci-dessous.

Lorsque le nombre seuil défini par la variable NbCarLire de caractères qui arrivent dans le buffer de réception du port est atteint, le « thread » appelle la fonction InComm(NbCarLire) du DLL et charge les caractères reçus dans un tableau dont la dimension varie en fonction de la taille de l'EPROM .

Au moment où toutes les données de l'EPROM sont lues, les tâches du thread sont suspendues pour libérer le processeur du processus de lecture afin qu'il puisse exécuter d'autres instructions.

VI.2.4) Ecriture de données

Contrairement à la lecture, l'écriture d'un caractère est faite en dehors du thread avec la fonction OutComm(char car) du DLL. Cette fonction fait appel à la fonction win32 nommée TransmitCommChar et prend comme paramètre le caractère à envoyer.

Remarque :

La manipulation d'un « thread » est similaire à la manipulation d'un port, après chaque création et à la fin d'une session, sa tâche doit aussi être arrêtée avec la fonction terminate( ).

VI-2-5) Organigramme de fonctionnement de l'ensemble

Le fonctionnement de l'interface peut être résumé avec l'organigramme ci-dessous.

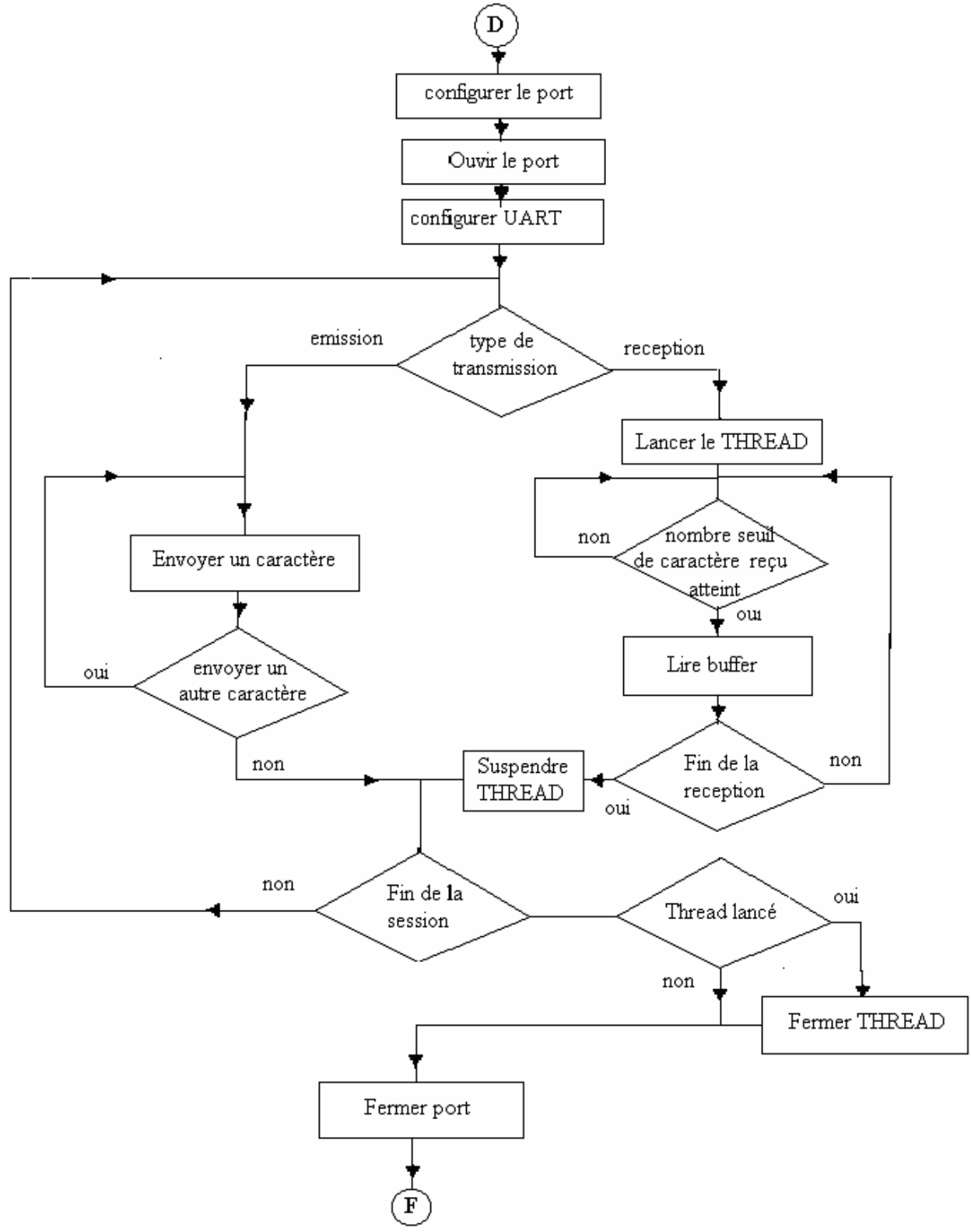


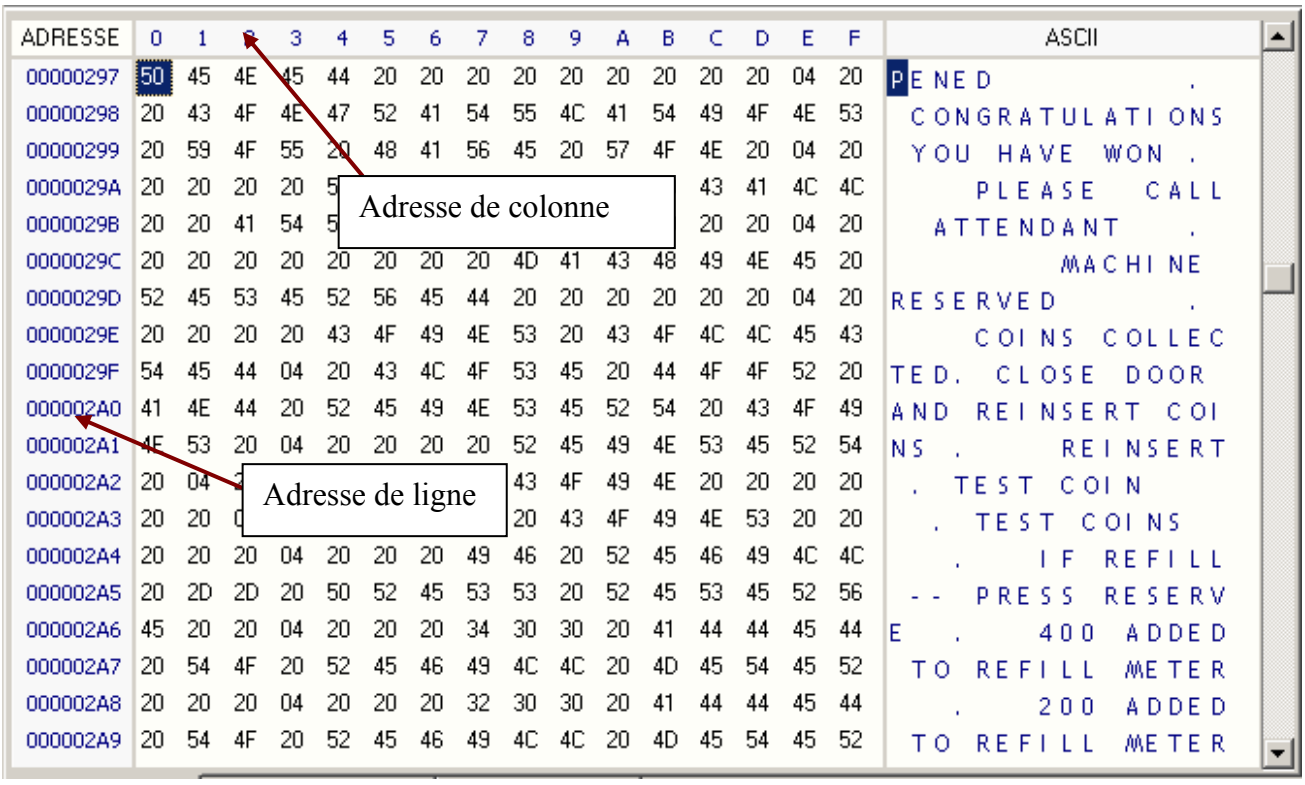
Figure VI.2 Organigramme de fonctionnement de l'interface

VI.3) Vérification et présentation des résultats

Avant de présenter les données résultant d'une lecture, elles sont tout d'abord rassemblées dans un pointeur de caractères nommé *Buf. Après cela, on relance une deuxième fois la lecture de l'eprom et cette fois ci, les données sont rangées dans un pointeur temporaire de même type et de même taille que précédemment nommé *Tmp.

La vérification de l'intégrité de données ainsi obtenues se manifeste par la comparaison entre les éléments des deux pointeurs.

Une fois que tout ceci est terminé, l'espace réservé au pointeur Tmp est libéré tandis que les données obtenues dans *Buf sont présentées dans un éditeur (Figure VI.3) sous forme hexadécimale et ascii (annexe III).



ADRESSE	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ASCII
00000297	50	45	4E	45	44	20	20	20	20	20	20	20	20	20	04	20	PENED .
00000298	20	43	4F	4E	47	52	41	54	55	4C	41	54	49	4F	4E	53	CONGRATULATIONS
00000299	20	59	4F	55	20	48	41	56	45	20	57	4F	4E	20	04	20	YOU HAVE WON .
0000029A	20	20	20	20	5								43	41	4C	4C	PLEASE CALL
0000029B	20	20	41	54	5								20	20	04	20	ATTENDANT .
0000029C	20	20	20	20	20	20	20	20	4D	41	43	48	49	4E	45	20	MACHINE
0000029D	52	45	53	45	52	56	45	44	20	20	20	20	20	20	04	20	RESERVED .
0000029E	20	20	20	20	43	4F	49	4E	53	20	43	4F	4C	4C	45	43	COINS COLLEC
0000029F	54	45	44	04	20	43	4C	4F	53	45	20	44	4F	4F	52	20	TED. CLOSE DOOR
000002A0	41	4E	44	20	52	45	49	4E	53	45	52	54	20	43	4F	49	AND REINSERT COI
000002A1	4E	53	20	04	20	20	20	20	52	45	49	4E	53	45	52	54	NS . REINSERT
000002A2	20	04	2						43	4F	49	4E	20	20	20	20	. TEST COIN
000002A3	20	20	0						20	43	4F	49	4E	53	20	20	. TEST COINS
000002A4	20	20	20	04	20	20	20	49	46	20	52	45	46	49	4C	4C	. IF REFILL
000002A5	20	2D	2D	20	50	52	45	53	53	20	52	45	53	45	52	56	-- PRESS RESERV
000002A6	45	20	20	04	20	20	20	34	30	30	20	41	44	44	45	44	E . 400 ADDED
000002A7	20	54	4F	20	52	45	46	49	4C	4C	20	4D	45	54	45	52	TO REFILL METER
000002A8	20	20	20	04	20	20	20	32	30	30	20	41	44	44	45	44	. 200 ADDED
000002A9	20	54	4F	20	52	45	46	49	4C	4C	20	4D	45	54	45	52	TO REFILL METER

Figure VI.3 Allure de l'éditeur hexadécimal-ASCII

Sur cet éditeur, le chiffre hexadécimal qui se trouve sur l'adresse formée par l'intersection de l'adresse ligne et de l'adresse colonne représente le caractère ASCII mémorisé dans le cellule mémoire de l'EPROM pointée par la même adresse.

VI.4) Sauvegarde des résultats et chargement de données

VI.4.1) Sauvegarde des résultats

Une fois que les données lues sont présentées sur l'éditeur, elles sont prêtes à être enregistrer dans un fichier d'extension *.EPD qui signifie EPROM data.

VI.4.2) Chargement de données

Pour charger les données à écrire dans un EPROM vierge dans l'éditeur Héxadécimal-ASCII, deux façons peut être utilisés :

- soit les données sont chargées à partir d'un fichier EPD existant
- soit on les saisit directement sur l'éditeur ASCII de l'interface

VI.5) Portabilité de l'application

Pour faciliter le transport du logiciel d'un ordinateur à un autre, l'exécutable et les fichiers DLL de l'application sont compressés dans un fichier installateur SETUP.EXE généré avec le logiciel InstallShield Express de Borland.

CHAPITRE VII RESULTATS ET DISCUSSIONS

VII.1) Résultats obtenus

VII.1.1) Interface matérielle

L'interface matérielle (figure VII-1) fonctionne sous 5V et consomme une puissance de 2,5W.

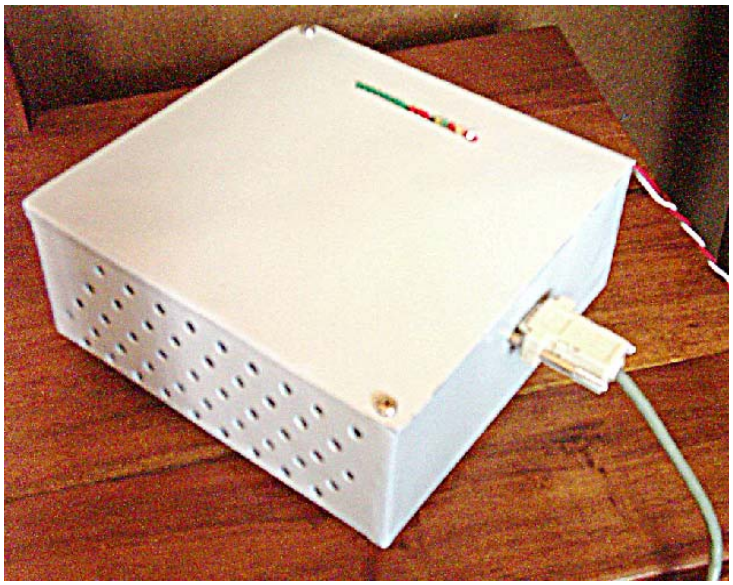


Figure VII.1 Aspect de la réalisation

VII.1.2) Interface logicielle

- Le logiciel d'acquisition peut être utilisé sur des machines pourvues des systèmes d'exploitation suivants :

- windows 95
- windows 98
- windows 2000 et NT
- windows XP

- La taille du fichier SETUP.EXE obtenu est de 3.2 Mo.
- L'interface logicielle comprend deux fenêtres principales :

- la fenêtre des résultats
- la fenêtre de configuration

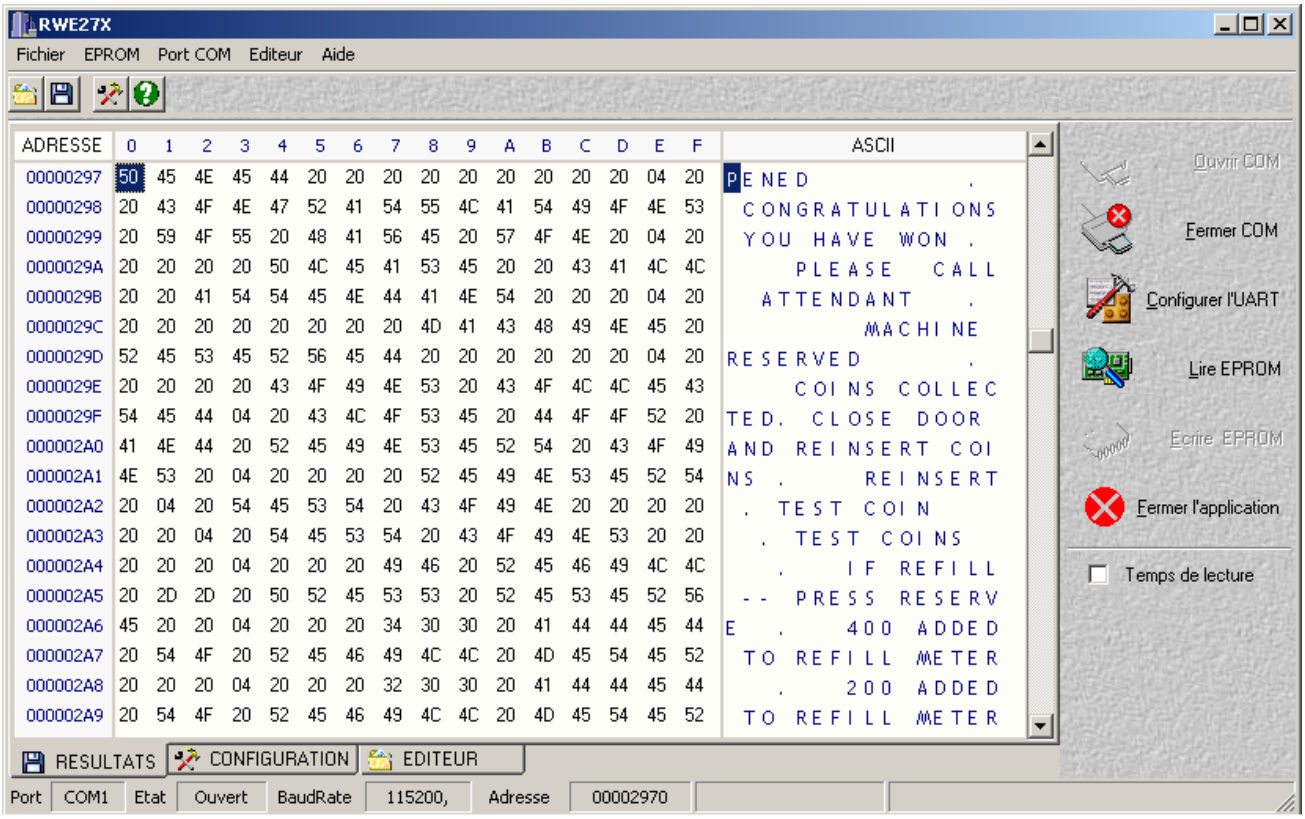


Figure VII.2 Aspect de la fenêtre des résultats

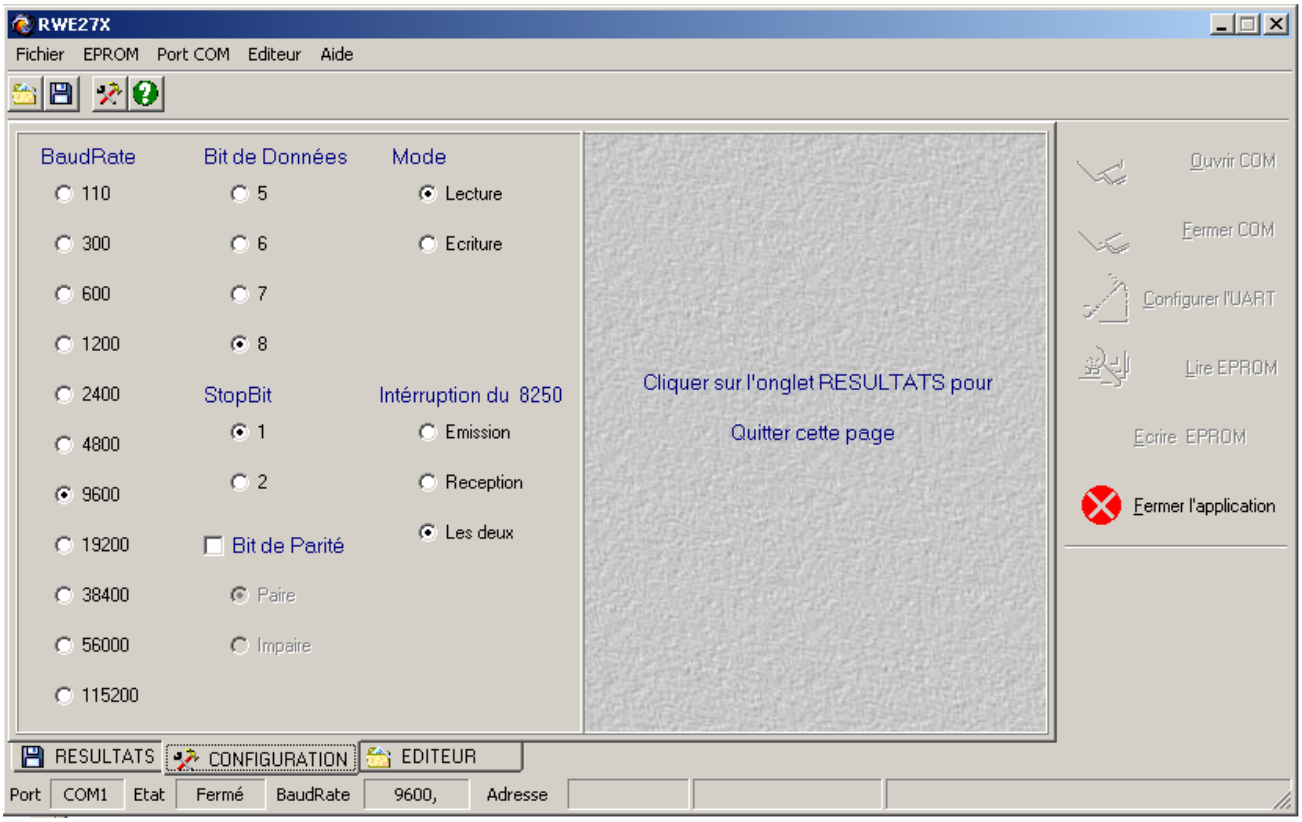


Figure VII.3 Aspect de la fenêtre de configuration

VII.1.3) Temps de lecture

Les temps les plus rapides obtenus avec un débit de 115200bps pour les 4 EPROMs sont :

Tableau VII.1 Temps de lecture en seconde

BAUDRATE	2764	27128	27256	27512
115200bps	0.83	1.61	3.17	6.30
	TEMPS (en seconde)			

VII.1.4) Temps d'écriture

Avec un débit de 115200bps et une impulsion d'écriture de 5ms, nous avons obtenus 7mn 21s pour écrire sur un EPROM 27512.

VII.2) Interprétation et discussions

VII.2.1) Côté matériel

Il a été mentionné précédemment que le circuit intégré 8250 est un UART conçu pour fonctionner avec un microprocesseur. Mais, à cause de l'inexistence de ce microprocesseur, nous avons été obligé de concevoir une carte capable de le remplacer.

En effet, dans ce travail, le 8250 n'est pas exploité au maximum car il existe d'autres fonctions du circuit intégré (à part la transmission de données) que nous n'avons pas utilisé comme la détection des erreurs qui peuvent se produire au cours d'un transfert.

VII.2.2) Coût de la réalisation

r. Par rapport aux travaux antérieurs concernant le même sujet utilisant un CDP 6402, la réalisation de l'interface présenté dans ce travail demande beaucoup plus de temps et de circuits intégrés. Par conséquent, le coût de la réalisation élevé et l'interface consomme beaucoup plus de puissance.

VII.2.3) Côté logiciel

En général, pour avoir plus de confiance sur les résultats issus d'une opération de lecture, le logiciel doit détecter à chaque arrivée de caractères l'erreur correspondant à la trame reçue comme :

- l'erreur de parité
- l'overrun

- l'état des buffers de réception ou transmission (overflow)

Or, dans le DLL, il n'existe pas encore de fonctions qui servent à détecter ces événements. Par conséquent, des erreurs peuvent se produire sans être remarquées.

Malgré cela, nous avons constaté au cours des essais que si l'UART est configuré correctement et que les circuits contrôleur de flux entre le programmateur et l'interface fonctionnent bien, les données reçues au cours d'une lecture sont valides.

Pourtant il est important de noter qu'une mise à jour du DLL est nécessaire pour compléter les fonctions manquantes.

VII.2.4) Temps de lecture

La performance de l'interface varie en fonction du débit utilisé pour chaque transfert.

En général, le débit D (ou « baud rate ») exprime le nombre de bits envoyés par seconde (Hausmann 1988). Ainsi, le débit maximal en caractère Q de l'interface peut être calculé avec la formule suivante :

$$Q = \frac{D}{N + P + S + 1}$$

où

- N : est le nombre de bits qui constitue un caractère
- P : est le nombre de bit de parité
- S : représente le nombre de bits de stop

Sur les résultats présentés dans le tableau VII.1, les caractéristiques de la trame de données utilisée étaient :

- 1 bit de début (d = 1)
- 2 bits de stop (S=2)
- un caractère composé 8 bits (N=8)
- aucun bit de parité (P=0).

Ce qui donne pour un débit D=115200bps un débit maximal en caractère Q=10472,73 caractères par seconde. Les temps théoriques nécessaires pour lire le 4 EPROMs ci-dessus sont donc :

Tableau VII.2 Temps de lecture théorique en seconde

BAUDRATE	**2764**	**27128**	**27256**	**27512**
115200bps	0.78	1.56	3.12	6.25
	TEMPS (en seconde)			

Comparé aux valeurs de ce tableau, nous constatons que les valeurs obtenues au tableau VII-2 sont légèrement supérieures de quelques millisecondes (de l'ordre de 0.05s).

Ceci provient du fait que le débit en caractère du programmateur utilisé au cours des essais était inférieure au débit maximal calculé (Tableau VII.2). Par conséquent, on peut dire que les temps de lecture obtenus sont satisfaisants.

VII.2.5) Temps d'écriture

Actuellement, avec les EPROMs récents, une impulsion d'écriture inférieure à 1ms peut être utilisée. En conséquence, la diminution de l'impulsion d'écriture du programmateur entraînera un gain de temps non négligeable pendant le gravage de l'EPROM.

Conclusion

L'interfaçage d'un appareil électronique sur un micro-ordinateur est fondé en général sur la correspondance entre deux entités de natures différentes. La première concerne les circuits électroniques de l'interface tandis que la seconde se tourne plutôt vers les programmations des différents modules qui gèrent son fonctionnement.

Dans cette étude, l'interface assure deux opérations :

- la lecture des données envoyées par le programmateur
- l'écriture des données dans un EPROM

Pour pouvoir établir la connexion entre ces deux appareils, plusieurs règles doivent être respectées :

- il y a d'une part les normes prédéfinies caractérisant le port de communication utilisé, qui a été le RS232C dans notre cas
- et d'autre part, il y a les caractéristiques de l'appareil à interfacer

De plus, le bon fonctionnement de la poignée de main gérant la liaison s'avère un outil essentiel pour réduire les pertes de données pendant un transfert.

Ce travail est basé sur l'utilisation du circuit intégré 8250 et du langage C++ orienté objet comme langage de développement, il démontre la faisabilité de concevoir un simple système d'interface avec l'UART 8250 sans recourir à un microprocesseur (difficile à trouver sur le marché local).

Les résultats obtenus au cours des essais ont été satisfaisants. De plus, l'interface peut être utilisée pour relier un appareil numérique 8 bits autre que le programmateur sur un ordinateur. Seulement, la performance de l'interface serait meilleure si un microcontrôleur ou un microprocesseur est utilisé pour piloter le 8250.

REFERENCES BIBLIOGRAPHIQUES

Biggerstaff T, 1989 : *Outils logiciels pour la programmation systeme*, *Microelectronic and computer technology corporation.* Masson Austin Texas, p 104-119

Cattoen M, 2003 : *Cours et travaux pratiques de microprocesseur*. Institut National Polytechniques de Toulouse, Département d'électronique et informatique

Dallas semicondoctor, *Fundamentals of RS 232 serial communication*, Application note 83

Data sheet isn 8250, 1997, *82C50A CMOS asynchronous communication element*, Intersil TM

Data sheet MAX 232, 1996, *RS-232 tranceivers,* Maxim, document téléchargeable sur le site http://www.maxim-ic.com

Davis P, 2003 : U*niversal asynchronous receiver transmitter VLSI design Lab,* Departement of Computer Science and Engineering University of South Carolina, revision v.04

Dowsing R, Woodhams F, 1987, *Principes de fonctionnement des ordinateurs,* Masson Paris

Duplessix A, 1997, *Etude et réalisation d'une liaison série a 1 Gbaud indépendante du codage de données,* thèse de doctorat de l'université de Paris VI

Hausmann P, 1988, *La bible du pc,* Micro application

Hirsch E., Wendling S, 1992, *Structure des ordinateurs,* Armand Collin

Huber B, 1998, *Microprocessor system data transfer interface design:an expert system approach using signal timing,* thèse de doctorat de l'université de Victoria.

Koren D, 1995, *Protocols and computer networks,* Cours en ligne de l'université de Tel-Aviv, *http://www2.rad.com/networks/1995/rs232/*

Leibson S, 1986, *Manuel des interfaces,* Mc Graw-Hill

Lilen H, 1977, *Introduction a la micro informatique du microprocesseur,* Radio Paris

Loukcs W, 2002, *Microprocessor systems and interfacing,* Waterloo university

Magarotto E, 2003 , *Support de cours de transmission et acquisition de données,* Laboratoire d'automatique et de procédés : ISMRA

Mercouroff, 1990, *Architecture matérielle et logicielle des ordinateurs et des microprocesseurs,* Armand Collin

Renesas application note, 2003, *Hardware interface technique for peripherals,* Ref: An0303018/Rev1.0

Sweet M, 1994, *Serial programming guide for posix operating system,* 5eme edition : 3ème révision, Appendix C GNU Free documentation license

Tisserant S, 2003 , A*rchitecture et technologies des ordinateurs,* ESIL

Viennet E, 1998, *Architecture des ordinateurs,* IUT de Villetaneuse département GPR

Vilotte P, Favreau P, *Architecture et organisation des ordinateurs une introduction,* article

ANNEXES

N° BROCHES			
DB 25	**DB 9**	**DENOMINATION**	**DESCRIPTION**
1		-	Earth Ground
2	3	T_x	Transmit Data
3	2	R_x	Receive Data
4	7	RTS	Request to Send
5	8	CTS	Clear to Send
6	6	DSR	Data Set Ready
7	5	GND	Ground
8	1	DCD	Data Carrier Detect
9		-	Reserved
10		-	Reserved
11		-	Unassigned
12		SDCD	Secondary DCD
13		SCTS	Secondary CTS
14		ST_x	Secondary T_x
15		DB	Transmit Clock
16		SR_x	Secondary R_x
17		DD	Receive Clock
18		-	Unassigned
19		SRTS	Secondary RTS
20	4	DTR	Data Terminal Ready
21		SQD	Signal Quality Detect
22	9	RI	Ring Indicator
23		DRS	Data Rate Select
24		DA	Auxiliary Clock
25		-	Unassigned

Dénomination des pins du DB9 et du DB25

1 Earth Ground

C’est une ligne optionnelle généralement mise à la terre de l’équipement principal pour protéger l’ensemble d’un éventuel choc électrique accidentel.

2 Transmit Data T_x

Les données séries sont émises sur cette ligne bit par bit. La condition de marquage impose qu'au moment où aucun caractère n'est envoyé sur T_X, son niveau doit être ramené à 1

3 Receive Data

Le terminal reçoit les données venant du DCE par cette ligne. De la même façon, le DCE doit remettre le niveau de R_D à 1 si aucun caractère n'est envoyé.

4 Request To Send

Le DTE active ce signal pour demander à DCE l'autorisation d'émettre un caractère

5 Clear To Send

Suite à une émission de RTS, DCE active ce signal pour informer le DTR de sa disposition à recevoir le caractère.

6 Data Set Ready

La liaison entre les deux équipements est établie si ce signal est égale à 1

7 Ground

C'est le signal pris comme référence pour les autres signaux.

8 Data Carrier Detect

DCD est utilisé par DCE pour aviser DTR que la trame de caractère a été reçue correctement.

12 Secondary Data Carrier Detect

Signal identique à DCD, il indique l'état de la réception d'une trame sur le deuxième canal de communication.

13 Secondary CTS

Signal de fonction analogue à CTS, utilisé pour le deuxième canal de transmission.

14 Secondary TXD

C'est la ligne d'émission de données pour le deuxième canal

15 Transmit Clock

DCE envoie un signal d'horloge par cette ligne afin que le terminal puisse synchroniser son circuit de transmission.

16 Secondary R_D

Ligne de réception des données séries pour le deuxième canal.

17 Receiver Clock

DCE envoie un signal d'horloge par ce ligne afin que le terminal puisse synchroniser son circuit de réception.

19 Secondary RTS

Signal de fonction analogue à RTS, utilisé pour le deuxième canal de transmission.

20 Data Terminal Ready

Quand ce signal est à l'état 1, le terminal informe qu'il est connecté à la ligne de transmission et prêt à recevoir les données de DCE.

21 Signal Quality Detect

DCE fixe ce signal à 1 si la trame qu'il a reçu est de bonne qualité.

22 Ring Indicator

Le RI est utilisé par un modem pour indiquer la réception d'un appel

23 Data Rate Select

Dans le cas où le DCE serait un modem à multiple débit, c'est par ce signal que le terminal choisit le débit utilisé pendant la transmission.

24 Transmit Clock

Le signal envoie un signal d'horloge vers le DCE par cette ligne, dans le cas où l'horloge serait du coté de DTE.

L'UART 8250 possède neuf registres internes de 8 bits pour gérer la transmission à part les deux registres de réception et de transmission de données. Sur ces neuf registres, seulement cinq sont utilisés et configurés dans ce travail, ce sont :

- Le registre de commande de ligne LCR
- Le registre de commande de modem MCR
- Le deux registres du diviseur de fréquence DLM et DLL
- Le registre d'interruption IER

1) Rôle de ces cinqs registres

a) Le registre LCR

LCR est le registre responsable de la définition du format de la trame asynchrone à envoyer ; la valeur introduit dans ce registre peut être lue et écrite et représente à la fois :

- le nombre de bits qui constitue un caractère : LCR (0) et LCR (1)
- le nombre de bit de stop qui compose la trame : LCR (2)
- l'utilisation ou non du bit de parité : LCR (3), LCR (4) t LCR (5)
- l'accès aux registres du diviseur de fréquence DLM et DLL. : LCR (7)

b) Le registre MCR

Ce registre sert en général d'interface à un modem, il possède cinqs signaux de commandes qui sont :

- DTR : MCR (0)
- RTS : MCR (1)
- OUT1 : MCR (2)
- OUT2 : MCR (3)
- LOOP : MCR (4)

Où seul le signal LOOP est en logique positive

c) Les registres DLL et DLM

Le diviseur de fréquence du 8250 est un diviseur de 16 bits subdivisé en deux registres de 8 bits qui sont DLL et DLM ; où DLL représente les huits bits de faible poids du diviseur et le DLM représente les huits bits restant.

Notons que l'accès à ces deux registres ne s'effectue qu'en mattant à un le huitième bit du registre LCR appelé DLAB

d) Le registre d'interruption IER

L'un des particularité du 8250 est la génération d'un signal d'interruption complexe en fonction des évènements qui se produisent. C'est par ce registre qu'on peut activer ou non le type d'interruption voulu ; et ces interruptions sont :

- L'interruption associé à la reception de données : IER (0)
- L'interruption associé à l'état du registre de tranmission : IER (1)
- L'interruption associé à l'état du ligne de transmission : IER (2)
- L'interruption associé à l'état du modem : IER (3)

2) <u>L'accès à ces registres</u>

L'accès à ces registres est identique à l'accès à un point mémoire par l'intermédiaire de l'introduction de l'adresse du registre cible sur les pins A_0, A_1 et A_2 du 8250. Ces adresse sont :

DLAB	**A_2**	**A_1**	**A_0**	**Registre**
0	0	0	1	IER
X	0	1	1	LCR
X	1	0	0	MCR
1	0	0	0	DLL
1	0	0	1	DLM

3) <u>L'entête du code de configuration utilisé dans le programme de configuration</u>

D'après ce qui a été dit au chapitre 7, l'entête du code de configuration commence à partir du $8^{ème}$ bit de la trame de configuration de 24 bits. Cet entête contient deux informations :

- l'adresse du registre cible
- le type d'opération à faire sur les registres comme la remise à zéro, écriture ou lecture.

Ces codes sont calculés en hexadécimal et présenté dans le tableau ci après. Sur ce tableau,

- les bits B_i représente les bits de commande utilisé dans la carte de configuration,
- les bits A_i représente les adresses des registres
- le signal CHARGER est le signal qui active la mémorisation des 24 bits désérialisés par les registres à décalage de la carte de configuration et assure leur tranfert vers la carte de tranmission. Par conséquent, le code de ce signal doit toujours être compris dans le code de configuration ; sinon rien ne se passe au niveau des registres.

	BIT	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	CODE HEXA
	NOM	B3	B2	B1	B0	-	-	CS	CG	-	-	WR	RD	MR	A2	A1	A0	
REGISTRES	RBR / THR / DLL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x000
	IER / DLM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0x100
	IIR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0x200
	LCR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0x300
	MCR	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0x400
OPERATION	RESET	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0x800
	LECTURE	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0x1000
	ECRITURE	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0x2000
	LOAD	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0x40000
	CONFIG	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0x10000
	SELECT	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0x20000
	CHARGER	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0xB00000
		BITS DE COMMANDE													ADRESSE D'UN REGISTRE			

Exemple

Pour écrire le chiffre X=15 (000111 en binaire et 0xF en hexadécimal) dans le registre de contrôle de données LCR, le code hexadécimal de l'information à envoyer est calculé de la façon suivante :

CODE = CHARGER+ LCR + SELECTION + CONFIG + ECRITURE + X

Qui donne :

CODE = 0xB00000 + 0x300 + 0x20000 + 0x10000 + 0x2000 + 0xF

CODE = 0xB3230F

SELECTION est introduit dans ce code afin de sélectionner le registre voulu ; tandis que, CONFIG sert à activer le transmetteur de bus de la carte de transmission en mode configuration.

Code	Caractère	Code	Caractère	Code	Caractère	Code	Caractère	Code	Caractère
0	[car. nul]	69	E	116	t	164	¤	211	Ó
…		70	F	117	u	165	¥	212	Ô
7	[sig. sonore]	71	G	118	v	166	¦	213	Õ
8	[ret. arrière]	72	H	119	w	167	§	214	Ö
9	[tabulation]	73	I	120	x	168	¨	215	×
10	[saut ligne]	74	J	121	y	169	©	216	Ø
11	[tab. vert.]	75	K	122	z	170	ª	217	Ù
12	[saut page]	76	L	123	{	171	«	218	Ú
13	[ret. chariot]	77	M	124	|	172	¬	219	Û
...		78	N	125	}	173	-	220	Ü
32	[espace]	79	O	126	~	174	®	221	Ý
33	!	80	P	…		175	¯	222	Þ
34	"	81	Q	128	€	176	°	223	ß
35	#	82	R	…		177	±	224	à
36	$	83	S	130	‚	178	²	225	á
37	%	84	T	131	*ƒ*	179	³	226	â
38	&	85	U	132	„	180	´	227	ã
39	'	86	V	133	…	181	µ	228	ä
40	(	87	W	134	†	182	¶	229	å
41	)	88	X	135	‡	183	·	230	æ
42	*	89	Y	136	ˆ	184	¸	231	ç
43	+	90	Z	137	‰	185	¹	232	è
44	,	91	[	138	Š	186	º	233	é
45	-	92	\	139	‹	187	»	234	ê
46	.	93	]	140	Œ	188	¼	235	ë
47	/	94	^	…		189	½	236	ì
48	0	95	_	142	Ž	190	¾	237	í
49	1	96	`	…		191	¿	238	î
50	2	97	a	145	‘	192	À	239	ï
51	3	98	b	146	’	193	Á	240	ð
52	4	99	c	147	“	194	Â	241	ñ
53	5	100	d	148	”	195	Ã	242	ò
54	6	101	e	149	•	196	Ä	243	ó
55	7	102	f	150	–	197	Å	244	ô
56	8	103	g	151	—	198	Æ	245	õ
57	9	104	h	152	˜	199	Ç	246	ö
58	:	105	i	153	™	200	È	247	÷
59	;	106	j	154	š	201	É	248	ø
60	<	107	k	155	›	202	Ê	249	ù
61	=	108	l	156	œ	203	Ë	250	ú
62	>	109	m	…		204	Ì	251	û
63	?	110	n	158	ž	205	Í	252	ü
64	@	111	o	159	Ÿ	206	Î	253	ý
65	A	112	p	160	[espace]	207	Ï	254	þ
66	B	113	q	161	¡	208	Ð	255	ÿ
67	C	114	r	162	¢	209	Ñ		
68	D	115	s	163	£	210	Ò		

Table des Codes ASCII et ASCII étendu

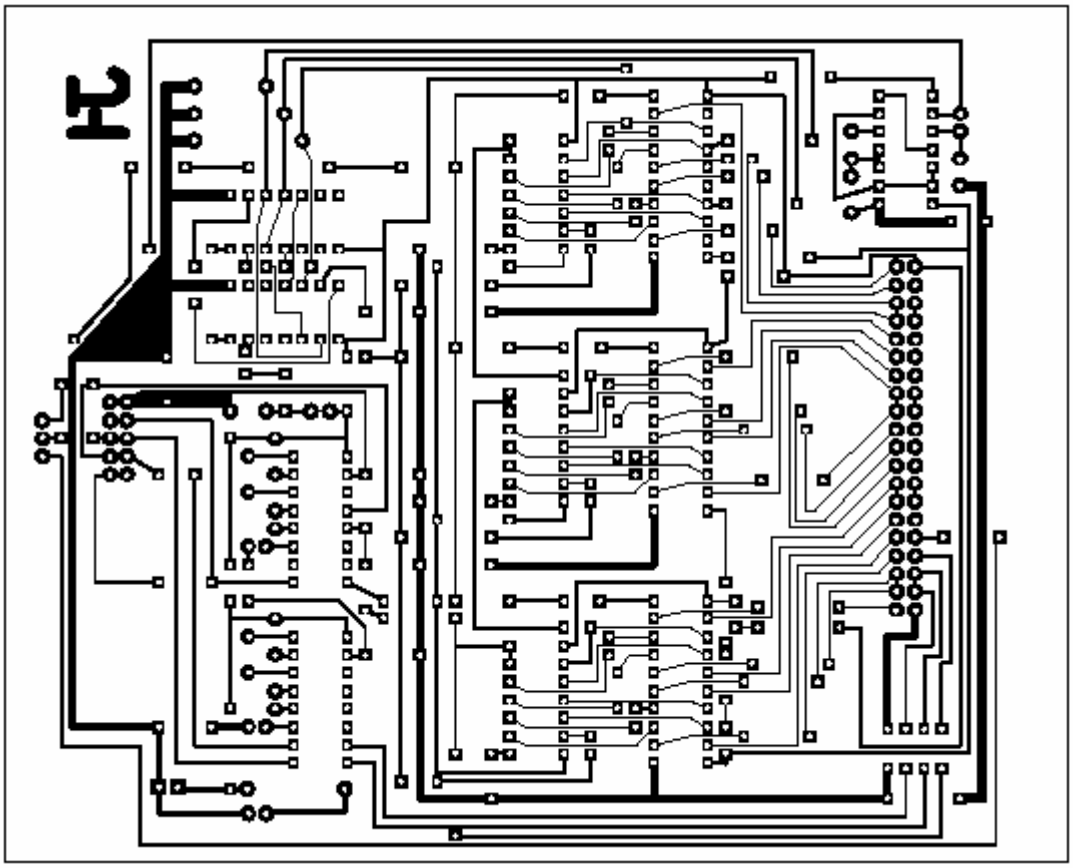


Figure 1 Tracé du circuit imprimé de la carte de configuration (vue côté composant)

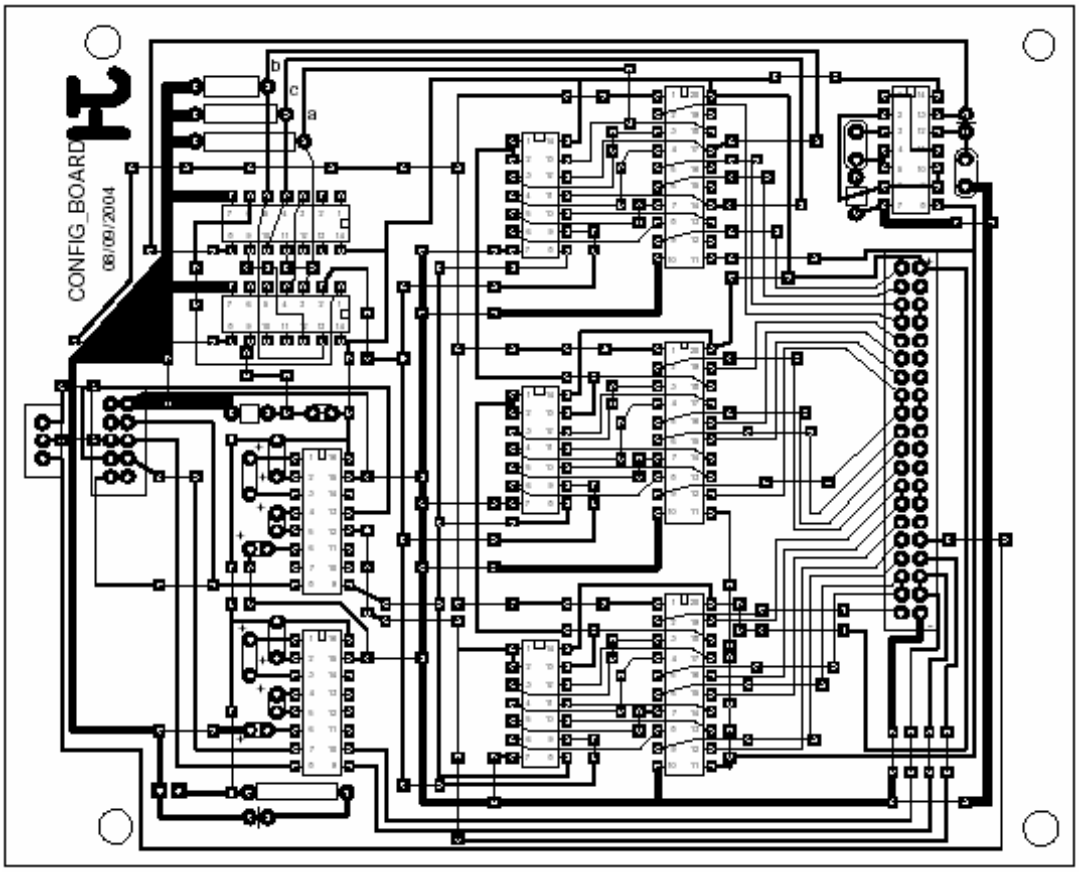


Figure 2 Implantation des composants

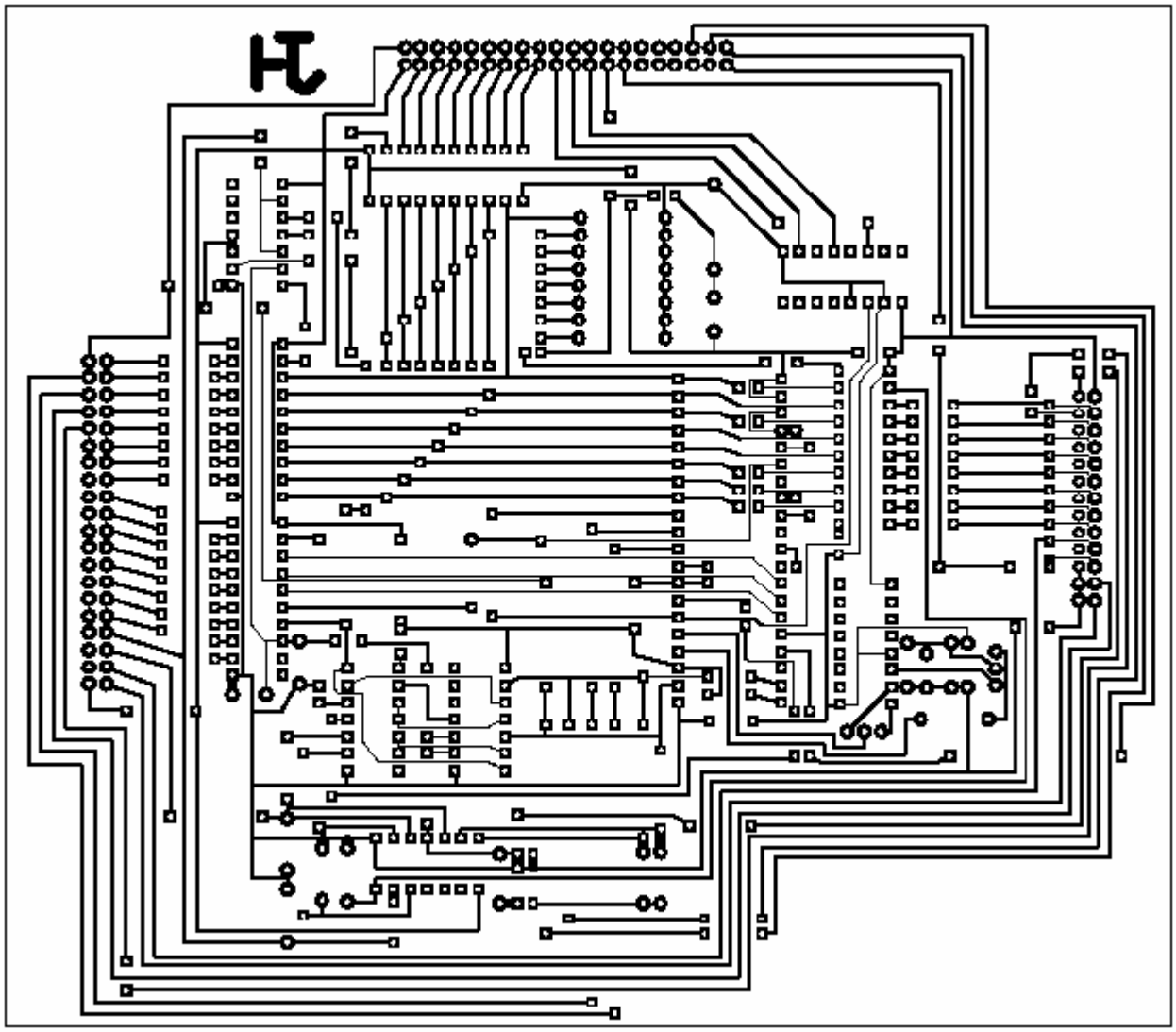


Figure 3 Tracé du circuit imprimé de la carte de transmission (vue côté composant)

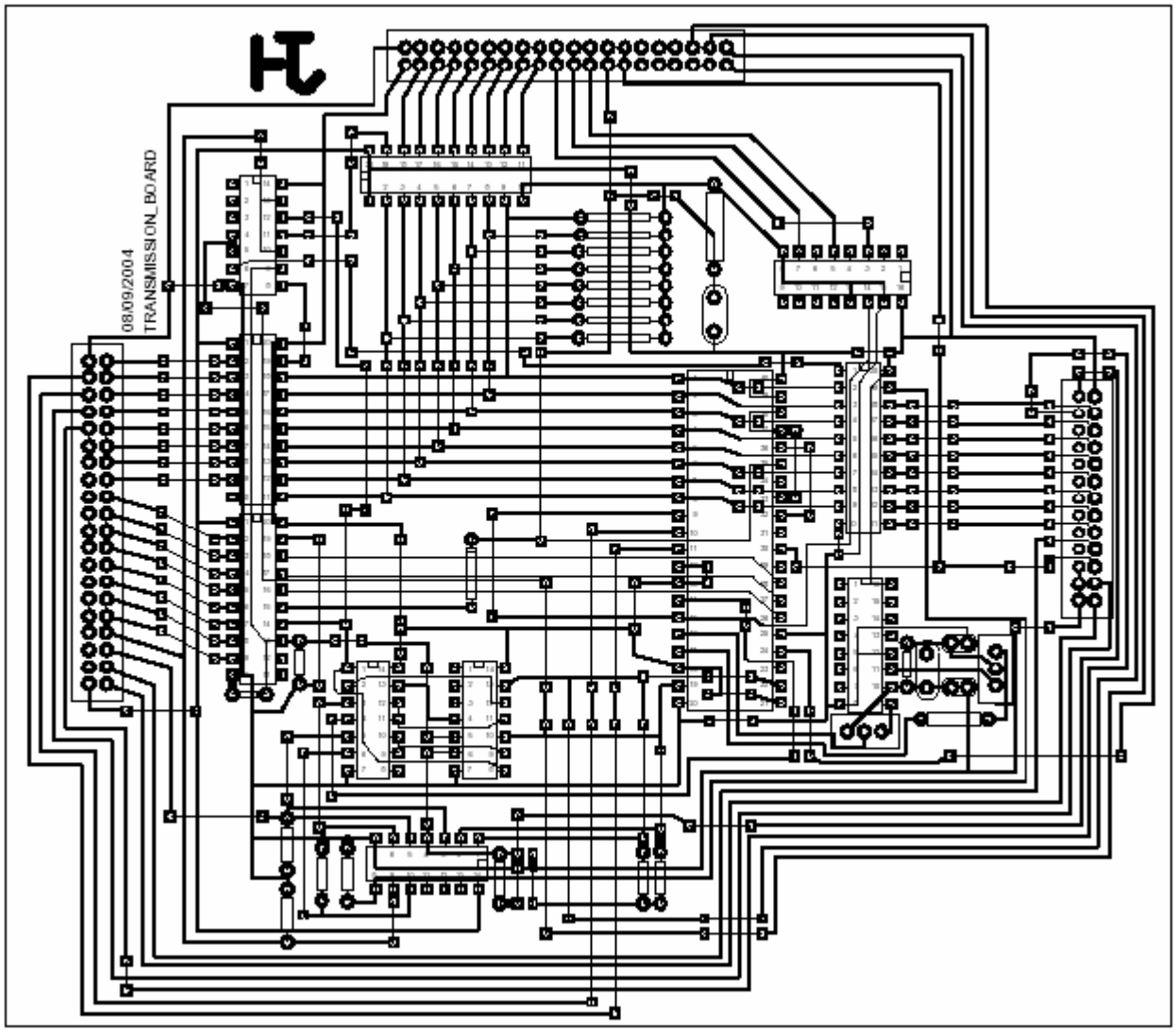


Figure 4 Implanatation des composants

CARTE DE CONFIGURATION			
Composant	Nombre	**Prixunitaire (Fmg)**	**Prix total (Fmg)**
Resistance ¼ W	6	500	3000
Condensateur chimique 1µF 16V	11	500	5500
Max 232	2	15000	30000
74LS00	1	15000	15000
74LS04	1	15000	15000
74LS11	1	15000	15000
74LS164	3	15000	45000
74LS273	3	15000	45000
Connecteur DB9	1	10000	10000
Connecteur Harting 40 broches	1	10000	10000
Support CI 14 broches	6	5000	30000
Support CI 16 broches	2	5000	10000
Support CI 20 broches	3	5000	15000
TOTAL			**248500**

CARTE DE TRANSMISSION			
Composant	Nombre	**Prix unitaire(Fmg)**	**Prix total(Fmg)**
Resistance ¼ W	19	500	9500
Condensateur chimique 100µF 10V	1	500	500
Condesateur céramique 22Pf	2	1000	2000
Quartz 1.8432 MHz	1	30000	30000
Diode signal	1	2500	2500
74LS00	1	15000	15000
74LS02	1	15000	15000
74LS04	1	15000	15000
74LS32	1	15000	15000
74LS245	4	15000	60000
CD4017	1	15000	15000
CD4060	1	15000	15000
8250	1	100000	100000
Connecteurs 40 broches	3	10000	10000
Support CI 14 broches	4	5000	20000
Support CI 16 broches	2	5000	10000
Support CI 20 broches	4	5000	20000
Support CI 40 broches	1	15000	15000
TOTAL			**369500 Fmg**

DIVERS			
Produit	Nombre	**Prix unitaire(Fmg)**	**Prix total(Fmg)**
Plaquette pré sensibilisée 30x30	1	120000	120000
Boîtier	1	50000	50000
Peinture	1	25000	25000
LED	17	1000	17000
TOTAL			**36500**

COUT TOTAL DE LA REALISATION	**830000 Fmg / 166000 Ar**

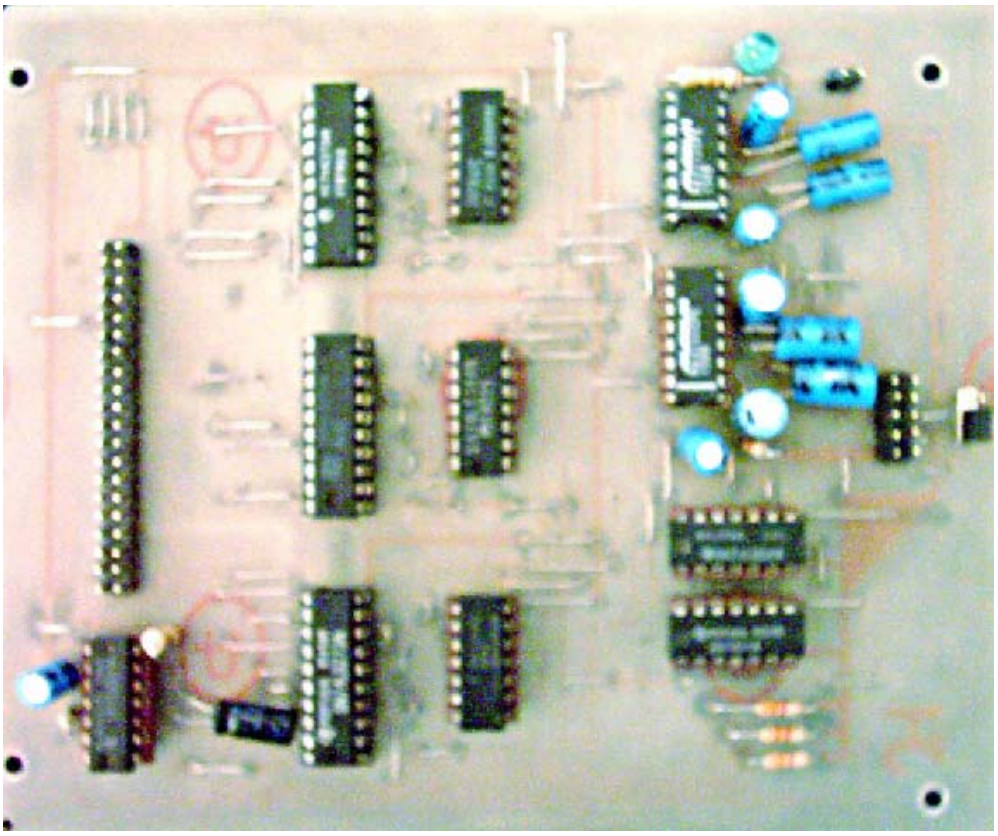


Figure 5 Aspect de la carte de configuration

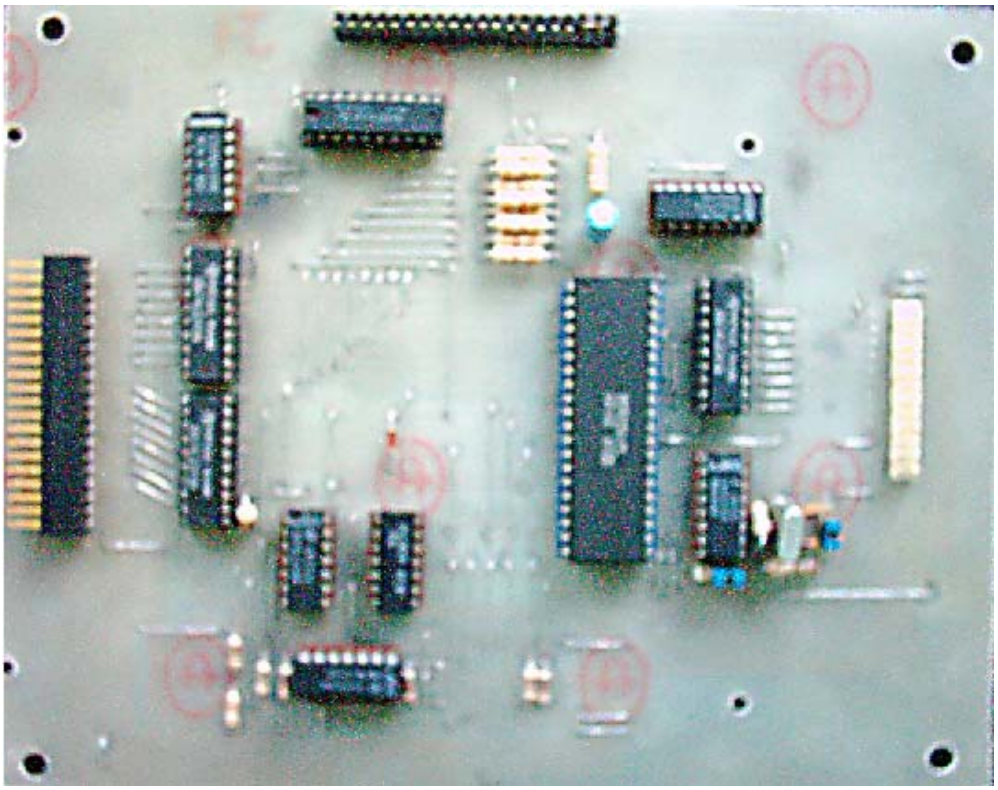


Figure 6 Aspect de la carte de transmission

Nom : ANDRIAMBELOSON
Prénom : Joely Andrianina
Adresse : B350D Anatihazo Antananarivo 101

Titre du mémoire : INTERFAÇAGE DU PROGRAMMATEUR D'EPROM RWE 27 SUR UN MICRO-ORDINATEUR POUR DES APPLICATIONS AUX APPAREILLAGES GEOPHYSIQUES

Résumé

Programmer des EPROMs est devenu un besoin dans les laboratoires de recherche actuels comme l'Institut et Observatoire de Géophysique d'Antananarivo puisqu'ils sont de plus en plus utilisés dans les appareils électroniques modernes.

L'interfaçage d'un programmateur d'EPROM sur un micro-ordinateur réduit le temps de programmation des mémoires. Par ailleurs, les opérations de lecture et d'écriture concernant ces mémoires sont facilitées.

La méthode suivie pendant la réalisation de cette interface est basée sur la programmation objet en ce qui concerne le logiciel d'acquisition. Quant à la transmission asynchrone de données, la norme RS 232 C a été utilisée pour le circuit d'acquisition.

Les données issues d'une opération de lecture peuvent être sauvegardées sur le micro-ordinateur sous forme de fichier.

Ce procédé peut être utilisé dans la maintenance d'un appareil à EPROM défaillant en rechargeant les contenus de cet EPROM dans un EPROM vierge.

Mots clés : EPROM, interface, micro-ordinateur, RS 232 C
Encadreur : M. RAMBOLAMANANA Gérard, Professeur
Responsable du Laboratoire de Sismologie, Sismique et Infrason
A l'Institut et Observatoire de Géophysique d'Antananarivo

Abstract

Today, programming EPROMs became a necessity at any laboratories since they used modern electronic devices as the Institute and Observatory of Geophysics of Antananarivo .

Therefore, interfacing an EPROM programmer on a computer seems to be the ideal tool to make faster and to automate the programming of these memories.

The approach followed during the realization of this interface is based on object programming for the software and using the standard R S232 C asynchronous transmission for the acquirement circuit.

The obtained data from the reading operation can be stored on a computer by saving them in a file.

This process can be used to maintain a device with faulty EPROM by reloading the contents of this EPROM in a virgin EPROM.

Key words : EPROM, interface, computer, RS 232 C