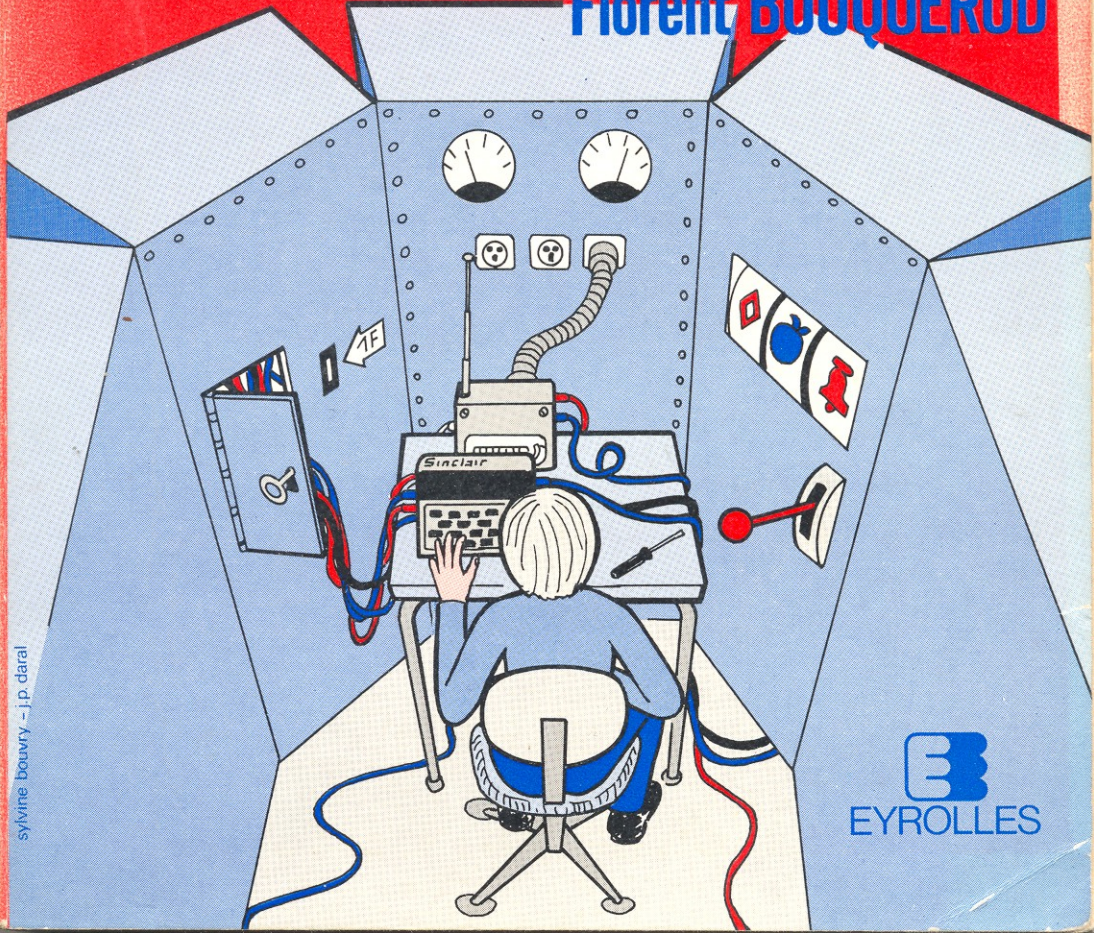
MICRO-ORDINATEURS

DES EXTENSIONS A CONSTRUIRE POUR VOTRE ZX 81

Florent BOUQUEROD



EYROLLES

sylvine bouvry - j.p. daral

Des extensions à construire pour votre ZX 81

DANS LA MÊME COLLECTION

SCHOMBERG	- Le Basic Universel.
SCHOMBERG	- Micro-ordinateurs : Comment ça marche ?
HERNANDEZ	- Pascal par l'exemple.
NOLLET	- La conduite du ZX 81.
PELLIER	- La conduite du TRS 80.
LADEVIE	- Votre gestion avec BASIC sur micro-ordinateur.
QUEINNEC	- Langage d'un autre type : LISP.
PELLIER	- Programmez vos jeux d'action rapde sur TRS 80.
ASTIER	- La conduite du l'APPLE II Tome 1 : le Basic de l'APPLE II Tome 2 : le système graphique et l'assembleur de l'APPLE II.
MONTEIL	- L'assembleur facile du 6502.
LEPAPE	- L'assembleur facile du Z 80.
OROS et	- ZX 81 à la conquête des jeux.
PERBOST	- CASSETTE - ZX 81 à la conquête des jeux.
DAX	- CP/M et sa famille.
NOLLET	- Langage machine, trucs et astuces sur ZX 81.
BICKING	- La conduite du PC 1211 (ou TRS 80 pocket).
TEJA	- Apprenez à parler à votre ordinateur.
MONTEIL	- La conduite du VIC 20.
PLOUIN	- La conduite de l'IBM-PC.
SAGUEZ	- Télécommande avec votre micro-ordinateur.
PELLIER	- Tout sur les disques du TRS 80 modèles I et III.
OROS et PERBOST	- La conduite du FX-702 P et FX-801 P.
GROS	- La conduite du PC 1500
HARWOOD	- Jeux et applications pour ZX SPECTRUM
HARTNELL	- Le grand livre du ZX SPECTRUM
VULDY	- Graphisme 3 D sur votre micro-ordinateur.
BOUQUEROD	- Des extensions à construire pour votre ZX 81
PINSON	- Le Basic en gestion sur Apple II
WILLARD	- La conduite du TI 99
DELANNOY	- Les fichiers en Basic... sur micro-ordinateur

Des extensions à construire pour votre ZX 81

par

Florent BOUQUEROD

Collection animée
par Richard SCHOMBERG

EYROLLES

61, boulevard Saint-Germain — 75005 Paris
1983

Introduction

Cet ouvrage est destiné aux personnes qui sont désireuses de pénétrer le monde de la micro-informatique. Il ne requiert aucune formation particulière et se veut accessible à tout individu, un tant soit peu passionné. Contrairement à un grand nombre d'autres livres de ce type, celui-ci dispense un bagage technique qui permet une compréhension approfondie des actions exécutées, mais aussi et surtout, il propose une série d'applications directes de ce qui vient d'être expliqué, avec tous les détails pratiques nécessaires à la réalisation.

Le matériel de base à partir duquel sont réalisées toutes les manipulations est le micro-ordinateur *** ZX 81 *** *de chez Sinclair, auquel on ajoutera, pour un investissement raisonnable, un ensemble de composants électroniques de type courant et facilement approvisionnables chez un fournisseur spécialisé local.*

Ce manuel ne fait en aucun cas double emploi avec toutes les publications parues sur le ZX 81 et qui ont inondé le marché, car il s'attache plus particulièrement à l'aspect matériel (ou Hardware suivant le terme consacré par les spécialistes) du micro-ordinateur sans négliger toutefois l'aspect programmation ou logiciel (ou Software pour les spécialistes); car l'adjonction d'un matériel plus performant à un système déjà existant n'a de sens en soi que si l'utilisateur est en mesure de le programmer habilement.

Cet ouvrage n'a pas la prétention de faire de chaque lecteur un micro-informaticien émérite, mais il peut néanmoins être un premier contact enrichissant et tout à fait pratique avec la conception et la réalisation des systèmes à microprocesseurs.

Dans cette optique le micro-ordinateur devient alors un outil éducatif qui permet de comprendre et de pratiquer les techniques microprocesseurs, et son utilisateur dépasse la fonction de programmeur ignorant tout du fonctionnement interne de la machine puisqu'il participe lui-même à l'extension matérielle de son système.

Ainsi la démarche suivie tout au long de cet ouvrage ressemble à s'y méprendre à celle qu'emploierait un micro-informaticien pour concevoir un équipement nouveau. Partant de données techniques délivrées par les constructeurs de composants celui-ci élabore un ensemble ou circuit électronique dont il dresse les plans détaillés puis passe à la réalisation ou cablage d'une maquette et enfin teste son fonctionnement.

C'est de cette manière que nous allons progressivement entrer dans ce monde de la micro-informatique qui se fait chaque jour de plus en plus envahissant dans notre vie quotidienne.

Cela peut aussi contribuer à la démystification de toutes ces technologies nouvelles qui laissent un peu rêveur la plupart d'entre nous et peut donner lieu à une réflexion plus intense sur les retombées d'une telle révolution mais aussi sur la puissance et les limitations des ordinateurs.

Le livre qui va suivre peut dès lors être décomposé en deux grandes parties:

— Dans un premier temps, nous reprendrons en nous appuyant sur l'exemple du micro-ordinateur ZX 81 l'architecture d'un système à microprocesseur en détaillant un à un les composants principaux et en donnant, au fil du texte, l'ensemble de la terminologie spécialisée indispensable à la compréhension de n'importe quel ouvrage de micro-informatique.

Cette partie peut, à la rigueur, être parcourue en diagonale par les lecteurs suffisamment avertis de ces questions et qui désireraient passer à l'aspect pratique plus rapidement.

— En effet, la seconde partie présente un ensemble de réalisations pratiques d'extensions du ZX 81, parmi lesquelles on compte:

— un générateur de notes,

— un chenillard à leds, programmable,

— une extension mémoire,

— etc...

Chaque partie concernant une extension se décompose elle-même en deux sous-parties fondamentalement distinctes:

— La première est une approche théorique des composants utilisés, elle comporte tous les détails utiles à leur mise en œuvre, et permettrait sans aucun doute à un lecteur désireux d'aller plus loin, de créer lui-même soit des variantes aux manipulations décrites, soit encore des extensions nouvelles basées sur ces mêmes composants.

— La seconde, beaucoup plus pratique, est proposée comme un exemple de fonctionnement parmi d'autres des composants cités plus haut. Depuis le montage, en passant par le test et jusqu'à la programmation, tout est là pour aider le lecteur dans sa progression. Cette partie lui permet de se familiariser à l'utilisation de ces composants complexes mais qui interviennent dans la plupart des applications des microprocesseurs.

Nous ne saurions trop conseiller au lecteur d'appréhender les parties théoriques sans trop insister sur la première lecture, puis d'y revenir lorsqu'il en sera à la réalisation pratique, pour éclaircir un certain nombre de points qui lui sembleront encore obscurs.

Table des matières

1

Présentation du matériel du ZX 81

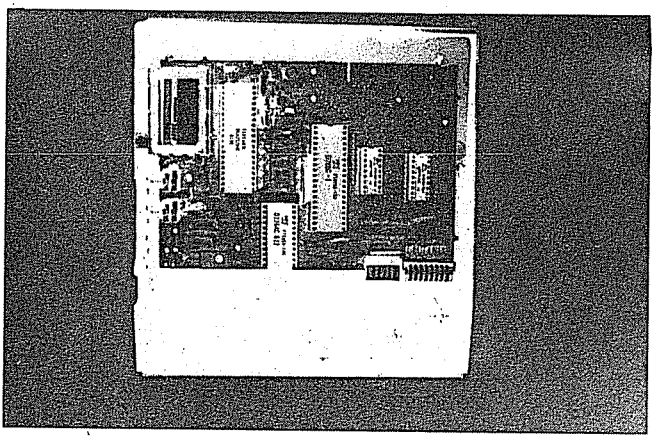
Donc vous êtes ou allez le devenir, un heureux possesseur de cette petite merveille que représente le micro-ordinateur ZX 81.

La documentation qui l'accompagne est très bien conçue pour l'utilisateur, seulement elle manque de détail en ce qui concerne le fonctionnement interne du micro-ordinateur.

Et avant de s'aventurer plus loin dans notre étude, il convient de bien cerner les différentes fonctions de chaque pièce constituant cet ensemble qu'est le ZX 81.

Nous allons donc analyser dans un détail suffisant à la compréhension des tâches à venir, chacun des composants principaux du micro-ordinateur à savoir:

— le microprocesseur,
— les mémoires; en distinguant la mémoire à lecture seule et la mémoire à accès aléatoire,
— le boîtier développé par Sinclair,
— le connecteur d'entrées-sorties réservé aux extensions,
— l'alimentation.



Intérieur du ZX 81

1.1. LE MICROPROCESSEUR

1.1.1. Description

Le microprocesseur du ZX 81 est un Z 80 (attention aux confusions entre ces deux appellations). Ce microprocesseur est assez largement utilisé dans les micro-ordinateurs actuellement sur le marché et disons encore qu'il est dans sa catégorie un des plus puissants. Il est fabriqué aux États-Unis à la fois par Zilog et Mostek, deux grands noms de la micro-informatique.

Physiquement ce microprocesseur se présente sous la forme d'un boîtier rectangulaire ou puce d'où sortent quarante broches de connexion. Celui-ci comporte une inscription du type : D 780C-1.

Sur le circuit du ZX 81 il occupe pratiquement la place centrale (voir photo ci-dessus).

1.1.2. Fonction

C'est ce composant qui détient en quelque sorte l'intelligence du système ce qui ne signifie pas qu'il est capable de prendre des initiatives par exemple mais qu'il peut réaliser une séquence d'actes très élémentaires et toujours les mêmes à la demande de son programmeur. Ce qui signifie aussi, en d'autres termes, qu'un microprocesseur ou plus généralement un micro-ordinateur n'a que l'intelligence de son programmeur.

Cette puce de silicium est très spécialisée, et à elle seule elle se comporte comme un véritable petit ordinateur ce qui justifie une autre désignation assez souvent utilisée: CPU pour Control Processing Unit.

Une telle machine est capable de suivre un programme préalablement mémorisé, ce programme est nécessairement sous la forme d'une séquence d'instructions en *code machine*, seul langage directement compréhensible au microprocesseur.

Les actions élémentaires dont sont capables les microprocesseurs sont essentiellement de type électrique, et même plus restrictivement encore, ils ne sont capables que de générer et de comprendre deux valeurs de tension électrique: le zéro volt et le + 5 V.

Ce sont ces deux valeurs de la tension électrique que l'on nomme les *niveaux logiques* et on a l'habitude de les représenter avec les chiffres 0 et 1:

0 pour le 0 volt ou masse électrique
1 pour le 5 volts

cela nous amène tout naturellement à l'utilisation d'un formalisme mathématique élémentaire: la numérotation en base deux ou numérotation binaire. Nous renvoyons les lecteurs qui ne seraient pas familiarisés à l'usage de ces notations à l'annexe 1, en fin d'ouvrage, ou il est expliqué en détail le calcul binaire.

La tâche du microprocesseur est alors, à partir des niveaux logiques présents sur ces broches d'entrées, de générer sur ces sorties des autres niveaux logiques qui pourront être exploités à leur tour par son environnement; c'est un cas typique de réalisation d'une instruction.

1.1.3. Les broches

Le Z 80 ne peut donc pas travailler tout seul, il doit avoir des liaisons avec les autres parties du système. Ces liaisons sont de trois types :

— les lignes de données,
— les lignes d'adresses,
— les lignes de contrôle.

Les lignes de contrôle sont généralement unifilaire, alors que les lignes de données sont rassemblées pour former le *bus donné* de huit pistes et que les lignes d'adresses forment le *bus adresse* de 16 pistes.

A) Le bus donnée (DATA BUS). Broches D0 à D7

Il est utilisé pour transporter 8 niveaux logiques en même temps et en parallèle. Cela représente les 8 informations qui, à un moment donné, transitent entre le CPU et et ses périphériques. Ces lignes peuvent servir soit à envoyer une donnée soit à en recevoir, on dit alors qu'il s'agit de *lignes bidirectionnelles.* Lorsque le bus donnée n'est pas utilisé par le CPU, celui-ci le met en état de *haute impédance,* ce qui signifie que le niveau logique présent sur le bus n'est alors pas significatif, et qu'un utilisateur extérieur peut éventuellement prendre le contrôle du bus pour communiquer avec les périphériques sans passer par l'intermédiaire du CPU.

B) Le bus adresse (ADRESS BUS). Broches A0 à A15

Ce bus-là contient aussi une information numérique codée sur 16 bits qui représente l'adresse du dispositif avec lequel le microprocesseur est en train de converser. Les lignes adresses sont essentiellement des sorties du CPU, elles peuvent elles aussi être mis en état haute impédance lorsque elles ne sont pas utilisées par le CPU.

Les adresses ainsi transportées vers tous les périphériques vont être analysées par ceux-ci, et si l'un d'entre eux reconnaît une adresse qui lui appartient il répondra aussitôt à la requête venant du CPU.

C) Les lignes de contrôle

Nous donnons une à une les différentes lignes de contrôle intervenant dans le brochage du microprocesseur. Il n'est pas très important de comprendre la fonction de chacune puisque certaines ne sont même pas utilisables dans le ZX 81 et d'autant plus que nous reverrons plus en détail et au moment voulu, celles dont nous nous servirons plus loin.

Disons encore que la plupart des signaux de contrôle sont *actifs à l'état bas*, cela signifie qu'il faut qu'on leur applique un niveau 0 (zéro volt) pour réaliser effectivement la fonction énoncée.

M1/ : Cycle machine 1, sortie active à l'état bas.
Cette ligne est une information que délivre le CPU pour indiquer qu'il est dans un cycle machine 1.
Nous ne donnerons pas plus de détail sur le cycle machine 1.

MREQ/ : Interrogation mémoire (MEMORY REQUEST), sortie active à l'état bas.
Cette ligne indique que l'adresse lisible sur le bus adresse est celle d'une mémoire et non pas celle d'un périphérique d'entrée ou de sortie.

IORQ/ : Interrogation entrée-sortie (INPUT/OUTPUT REQUEST), sortie actie à l'état bas.
Cette ligne indique que l'adresse lisible sur le bus adresse est celle d'un périphérique d'entrée ou de sortie et non pas celle d'une mémoire.

RD/ : Demande de lecture (READ), sortie active à l'état bas.
Cette ligne indique que le CPU demande à lire le contenu du périphérique adresse. Ce contenu se retrouve alors sur le bus données.

WR/ : Demande d'écriture (WRITE), sortie active à l'état bas.
Cette ligne indique que le CPU demande à écrire le contenu du bus données dans le périphérique adresse.

RFSH/ : Signal de rafraîchissement (REFRESH), sortie active à l'état bas.

Ce signal indique que les sept bits de poids faibles du bus adresses contiennent l'adresse de rafraîchissement de toutes les mémoires dynamiques.
Nous reviendrons plus en détail sur cette ligne au moment de l'interfaçage de mémoires dynamiques.

HALT/ : État d'arrêt du processeur, sortie active à l'état bas.
Ce signal indique que le CPU est dans un état de repos, et qu'il ne repartira qu'après réception d'un signal d'interruption quelconque.

WAIT/ : Attente, entrée active à l'état bas.
Ce signal indique au CPU qu'on désire ralentir sa vitesse d'exécution.

INT/ : Reconnaissance d'interruption, entrée active à l'état bas.
Ce signal de demande d'interruption de la tâche du CPU est envoyé par les équipements extérieurs qui nécessitent alors un traitement immédiat. Nous discuterons plus en détail tous ces problèmes tournant autours des interruptions.

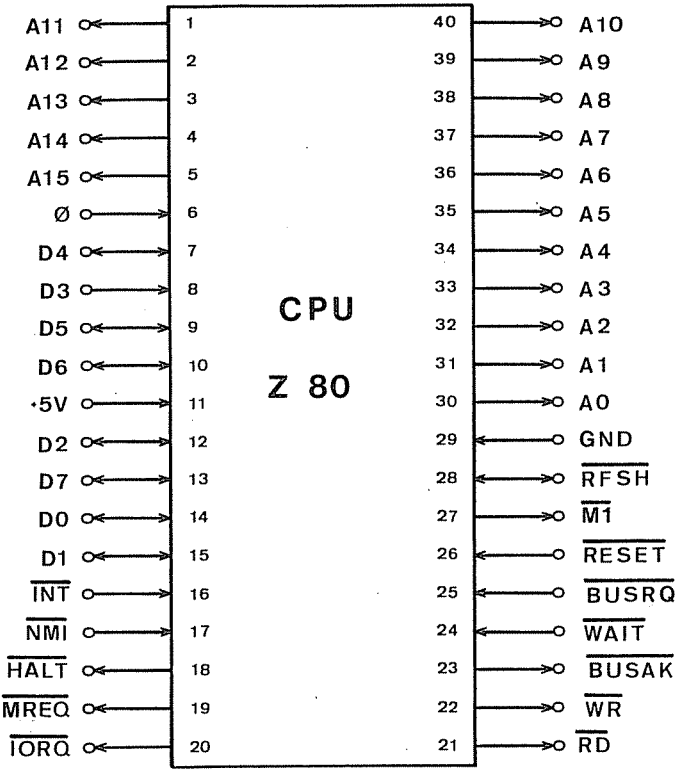
NMI/ : Interruption non-masquable (NON MASQUABLE INTERRUPT), entrée active à l'état bas.
Ce signal est aussi une demande d'interruption de la part des équipements extérieurs, mais contrairement à la précédente celle-ci ne peut pas être empêchée de manière logicielle.

RESET/ : Réinitialisation du processeur, entrée active à l'état bas.
Une intervention extérieure sur cette broche provoque la remise à zéro du compteur ordinal qui est chargé de compter les différentes instructions déroulées depuis le début d'une séquence que constitue un programme. Autrement dit le RESET fait repartir le microprocesseur depuis le début de son programme.

BUSRQ/ : Demande de mise à disposition du bus (BUS REQUEST), entrée active à l'état bas.
Ce signal demande de rendre disponible les bus du CPU pour qu'un autre équipement puisse les contrôler.

BUSAK/ : Accusé de réception du signal précédant (BUS ACKNOWLEDGE).
Sortie active à l'état bas.
Ce signal est en quelque sorte la réponse du CPU à la demande précédente (BUSRQ). Celui-ci signale qu'il vient de libérer ses différents bus (données et adresses) et que l'équipement demandeur peut les utiliser.

ϕ : Entrée d'horloge.
C'est par l'intermédiaire de cette entrée qu'un oscillateur externe délivre au CPU un signal périodique ou horloge destiné à séquencer les différentes tâches élémentaires exécutée par le microprocesseur.



1.1.4. La structure interne

La constitution interne du CPU est incroyablement compliquée, mais heureusement on peut diviser cette structure en cinq parties principales :

— *L'unité de contrôle* (UC) : Elle supervise tous les signaux qui entrent ou qui sortent du microprocesseur et cela uniquement en suivant les directives commandées par le programme.

— *Le registre d'instruction* : Un registre désigne en quelque sorte une mémoire spécialisée interne au microprocesseur lui-même. Il permet de retenir un court instant le contenu du bus des données lorsqu'il est sensé contenir un code opération.

— *Le compteur de programme* (PC) : Il s'agit d'un registre pair, c'est-à-dire mémorisant 16 bits. Il est utilisé pour conserver l'adresse de la mémoire qui contient l'instruction en cours d'exécution.

— *Les 24 registres utilisateurs* : Ils sont entièrement disponibles pour l'utilisateur. Nous en donnons la liste exhaustive sans plus d'explication :

A	: Accumulateur ou registre spécialisé qui entre en jeu dans les opérations logiques ou arithmétiques.
BC et DE	: Registres pairs à usage général.
HL	: Registre pair désignant plus particulièrement l'adresse de la mémoire sur laquelle on travaille.
F	: Registre des drapeaux ou flags, rassemble dans un octet les indicateurs binaires de condition.
IX et IY	: Registres pairs d'index.

L'ensemble des registres alternés : A', B', C', D', E', F', G', H remplacent dans certaines conditions particulières les registres primaires.

SP	: Pointeur de pile (STACK POINTER) permet de gérer une zone mémoire extérieure au CPU toute particulière nommée pile.
I	: Registre des vecteurs d'interruptions.
R	: Registre contenant l'adresse de rafraîchissement.

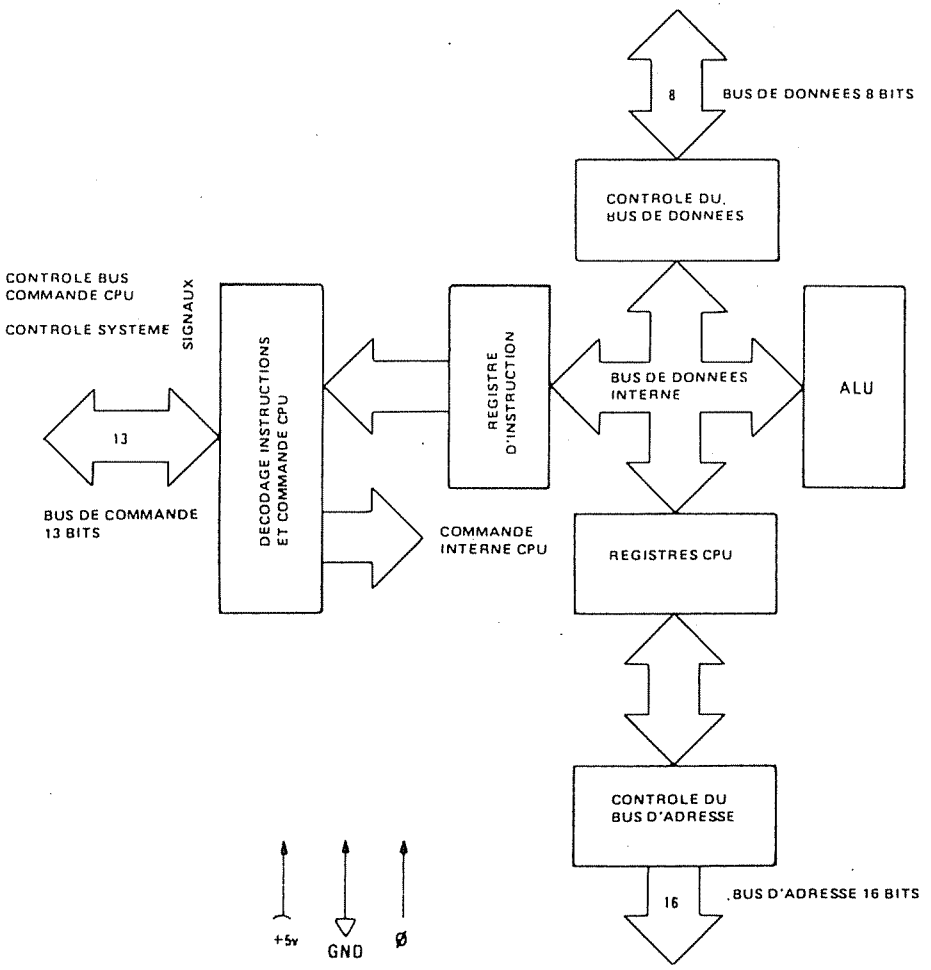
L'ensemble de ces registres est figuré dans le diagramme suivant :

Groupe principal		*Groupe alterné*		
Accumulateur A	Flags F	Accumulateur A'	Flags F'	
B	C	B'	C'	Registres d'usage général
D	E	D'	E'	
H	L	H'	L'	

Vecteur d'interruptions I	Compteur de Rafraîchissement R	
Registre d'index	IX	Registres d'usage particulier
Registre d'index	IY	
Pointeur de pile	SP	
Compteur de programme	PC	

— *L'unité arithmétique et logique* (ALU) : C'est ce dernier bloc fonctionnel qui réalise toutes les opérations arithmétiques ou logiques.

Cette représentation simplifiée de la structure interne est figurée dans le diagramme ci-après.



1.1.5. Le séquencement des opérations

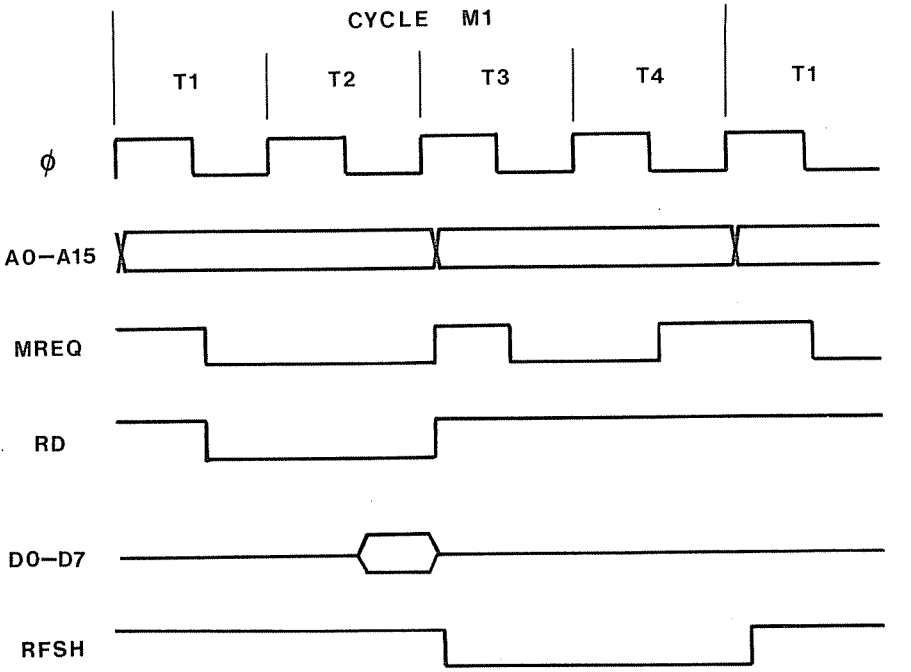
Une des limitations du microprocesseur est qu'il ne peut en aucun cas faire deux choses à la fois. Le passage d'une opération à la suivante est cadencé par une horloge, c'est-à-dire un système électronique oscillant, fournissant un signal périodique (alternativement aux niveaux logiques "0" et "1") et changeant d'état un certain nombre de fois par seconde, ce qui définit sa *fréquence.*

Sur le ZX 81 l'oscillateur est construit autours d'un cristal de céramique qui oscille à la fréquence (F) de 6.5. megaherz (MHz) ou millions de pulsations par seconde. Cette fréquence est ensuite divisée par deux pour obtenir une nouvelle horloge qui vient attaquer directement le microprocesseur. C'est celle-ci qui sert de référence temporelle et sa fréquence est : F = 3,25 MHz.

La durée d'une oscillation définit ce qu'on appelle la *période*, on la note souvent T est c'est aussi l'inverse de la fréquence : T = 1/F.

Une opération élémentaire accomplie par le microprocesseur peut durer entre 4 et 23 périodes d'horloge du CPU.

La représentation de l'état d'une ligne par un nombre binaire n'est en fait qu'un cliché instantanné du contenu de cette ligne, c'est parfois insuffisant et pour certaines analyses plus approfondies on utilise alors un diagramme temporel dans lequel sont représentés les niveaux successifs du signal en fonction du temps. On obtient un diagramme comme l'exemple ci-dessous appelé *timing* ou *chronogramme.*



Exemple de timing ou chronogramme

Remarquez sur cette planche la ligne supérieure qui représente le signal périodique de l'horloge et qui sert souvent de référence dans ce type de diagramme.

1.1.6. Le concept d'instruction en code machine

Nous l'avons déjà entrevu : ce sont les instructions du programme qui dictent au microprocesseur les actions qu'il doit mener. Parmi ces différentes opérations élémentaires exécutables par le CPU on peut citer pour l'exemple :

— le transfert de contenu entre deux registres,
— la lecture d'une mémoire ou d'un périphérique,
— l'écriture dans une mémoire ou un périphérique,
— les opérations logiques entre un registre et l'accumulateur,
— etc.

Le but d'un programme est alors de réaliser une tâche parfois compliquée, simplement en la décomposant en ces diverses opérations élémentaires.

Après avoir présenté le Hardware, le moment est venu de parler de la structure réelle d'un programme en code machine. Celui-ci est généralement écrit en *langage assembleur*, cela signifie que l'on a donné des *mnémoniques* (ensemble de lettres rappelant la fonction) à chacune des instructions pour rendre la lecture plus facile. Sinon un programme en langage machine apparaît simplement comme une liste de nombres binaires ou à la rigueur hexadécimaux. Soit par exemples :

Code hexadécimal	Mnémoniques	Signification
76H	HALT	Arrêt du processeur (HALT)
78H	LD A, B	Chargement (LOAD) de A par B
00H	NOP	Pas d'opération (NO OPERATION)

Les mots de la colonne signification et situés entre parenthèses sont les termes anglais qui ont servi à l'élaboration des mnémoniques.

Il y a une grande ressemblance entre un programme BASIC et un programme en langage machine dans ce sens que chaque ligne de langage machine doit comme en BASIC comporter une adresse. Cette

adresse est alors la valeur que doit prendre le compteur de programme lorsqu'il passe d'une instruction à la suivante.

La liste des instructions du Z 80 est donnée en annexe 2. Ajoutons quand même qu'un lecteur désireux de programmer en assembleur devra néanmoins trouver une documentation complémentaire sur ce langage. Nous ne la donnons pas ici car cela sort de notre objectif.

Néanmoins, nous conseillons à ce titre l'excellent livre issu de la même collection : « Langage machine, trucs et astuces sur ZX 81 ».

1.1.7. Le concept d'interruption

Encore une notion à connaître pour bien comprendre certains fonctionnements du microprocesseur : il s'agit des *interruptions* abrégées en IT.

En effet, imaginez que votre Z 80 est en train d'accomplir une tâche principale et que subitement un équipement périphérique nécessite d'urgence les services du CPU. Ce dernier peut alors être interrompu en lui envoyant sur une de ses deux entrées d'interruption (NMI/ et INT/), une impulsion négative.

A partir de cet instant le microprocesseur réalise la séquence suivante :

— il termine son instruction en cours,
— il regarde s'il doit tenir compte ou non de cette demande d'interruption. Sur le Z 80 l'IT INT/ est masquable, cela signifie qu'on peut par programme demander au CPU d'ignorer ou de considérer la requête d'interruption c'est le rôle des deux instructions spéciales :

DI (DISABLE IT)	masque l'interruption INT
EI (ENABLE IT)	valide l'interruption INT

Par contre l'IT NMI/ comme son nom l'indique n'est pas masquable et ne peut être ignorée par le microprocesseur quelque soient les circonstances.

— Dans le cas où l'interruption est reconnue, le compteur de programme est automatiquement positionné à :

38H pour NMI/
68H pour INT/

ce qui permet d'effectuer un branchement obligatoire dans un programme réservé au service de l'interruption.

L'adresse où on a quitté le programme principal a été préalablement sauvegardée sur la pile.

— Le service d'interruption se passe comme n'importe quelle autre routine puis lorsqu'elle est terminée, le CPU se rebranche là où il a été interrompu afin de reprendre sa tâche principale.

Voilà tout ce que nous dirons ici sur le microprocesseur Z 80, c'est de loin le composant le plus complexe du système et nous pourrions développer encore longtemps ce thème; mais ce n'est pas notre propos principal et nous pensons avoir rempli notre contrat qui est rappellons-le de fournir une base de renseignements permettant une bonne compréhension des manipulations à venir. Passons maintenant à une autre fonction très importante du système: la mémoire.

1.2. LES MÉMOIRES

Le microprocesseur ne peut en aucun cas travailler sans ses mémoires avec lesquelles il échange constamment des données. Ces données peuvent être de deux types:

— ou bien elles sont figées une fois pour toute pour un système donné; par exemple un programme moniteur, ou une série de messages parfaitement immuables,

— ou bien elles peuvent être modifiées en cours d'un traitement quelconque.

Cette distinction nous introduit dès à présent les deux grandes familles de mémoires:

— les *mémoires à lecture seule* (READ ONLY MEMORY, ROM) qui contiennent la plupart du temps le programme et un certain nombre de données immuables et qui peuvent comme leur nom l'indique, seulement être lues. De plus on peut couper l'alimentation d'un tel dispositif sans pour autant perdre leur contenu ceci explique qu'on appelle aussi parfois ces composants des *mémoires mortes,*

— les *mémoires à accès aléatoire* (RANDOM ACCES MEMORY, RAM) qui contiennent en général des données à caractère modifiable suivant le déroulement du programme et qu'on peut accéder soit en lecture soit en écriture, c'est-à-dire soit pour écrire des bits qu'on a l'intention de retrouver plus tard soit pour lire ces bits qu'on avait stockés précédemment. Lorsque qu'on coupe l'alimentation d'un tel dispositif son contenu est perdu à tout jamais, cela leur justifie une seconde appellation : celle de *mémoire vive.*

Mais voyons cela plus en détail en prenant pour exemples les composants du ZX 81.

1.3. LA ROM

1.3.1. Fonction

C'est elle qui contient toute la partie *programme-système* ou encore *logiciel résident,* appelé ainsi parce qu'il n'est pas besoin de le charger soit par le clavier soit par le magnétophone avant de pouvoir demander son exécution.

Sur ce point de vue, le logiciel système s'oppose au logiciel utilisateur qui représente en fait tous les programmes BASIC ou assembleur que le lecteur a écrit lui-même.

1.3.2. Dans le ZX 81

La ROM occupe les adresses de 0 à 2000H ou de 0 à 8196 en décimal soit encore 8 kilo-octets de mémoires sachant que 1 kilo-octet représente 2 ** 10 = 1 024 octets.

Cette partie est donc à lecture seule et il est très facile de le vérifier en exécutant les commandes BASIC :

```
PRINT PEEK 0 (demande de lecture de la position mémoire adressée
en 0, en début de la ROM)
```

on obtient :

211

et POKE 0,0 (demande d'écriture d'un zéro dans la mémoire adressée en 0) puis à nouveau :

PRINT PEEK 0

On obtient toujours 211, ce qui signifie que l'opération d'écriture précédente n'a pas fonctionné avec ce dispositif comme la ROM.

Le logiciel contenu dans la ROM ZX 81 a plusieurs fonctions :

— il y a d'abord les routines d'initialisation et d'autotest de la configuration du système,

— puis c'est l'ensemble des routines d'entrées-sorties, c'est-à-dire les sous-programmes qui font la liaison entre le microprocesseur et ses périphériques : écran, magnétophone, clavier...,

— il y a ensuite l'interpréteur BASIC qui fait une analyse syntaxique de chaque ligne entrée pour savoir si elle a une forme correcte, puis lorsqu'on commande l'exécution, transforme cette ligne en code reconnu par la machine afin que le microprocesseur l'exécute.
Il contient, de plus, l'ensemble des fonctions spéciales, mathématiques, logiques, etc...

— Enfin dans les adresses hautes de la ROM, entre 1E00H et 1FFFH soit 7680 et 8191 en décimal, se situe le générateur de caractères, zone ou sont codés en langage machine les motifs destinés à représenter les différents caractères du clavier sur l'écran. L'exemple ci-contre montre de quelle manière on représente un caractère dans la ROM.

EXEMPLE DU "A"

	Codages			*Affichage sur écran*
Adresses	Binaire	Hexa	Décimal	
7984	0 0 0 0 0 0 0 0 =	OOH =	0	. . . . .
7985	0 0 1 1 1 1 0 0 =	3CH =	60	. . * * * * . .
7986	0 1 0 0 0 0 1 0 =	42H =	66	. * . . . . * .
7987	0 1 0 0 0 0 1 0 =	42H =	66	. * . . . . * .
7988	0 1 1 1 1 1 1 0 =	7EH =	126	. * * * * * * .
7989	0 1 0 0 0 0 1 0 =	42H =	66	. * . . . . * .
7990	0 1 0 0 0 0 1 0 =	42H =	66	. * . . . . * .
7991	0 0 0 0 0 0 0 0 =	OOH =	0	. . . . .

1.3.3. Les broches

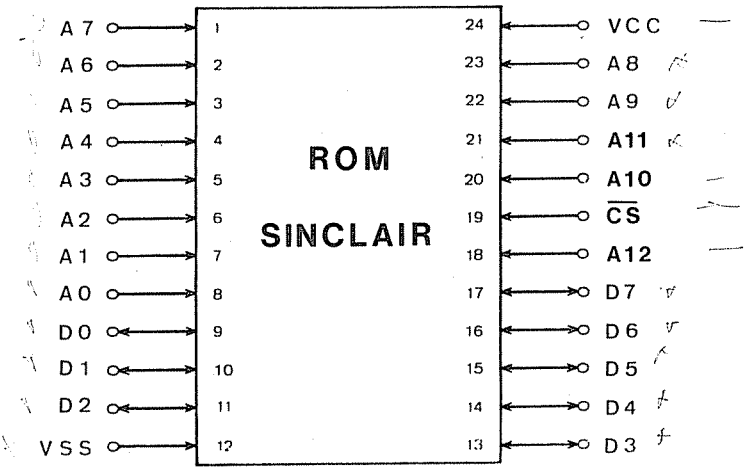
Le brochage de la ROM BASIC Sinclair est donné dans les lignes suivantes. On retrouve :

— le bus de données dans son intégralité,

— une partie du bus adresses, de A0 à A12 soit au total 13 lignes d'adresses qui permettent de repérer 2**13 = 8 192 positions mémoires différentes,

— les broches d'alimentation, 0 et + 5 V,

— une broche CS/ ou Chip Select barre qui permet, lorsque la tension est à zéro sur cette patte de circuit d'indiquer que le microprocesseur est en train de dialoguer avec ce boîtier plutôt qu'avec les autres du système.



1.3.4. La famille ROM au complet

Mais un lecteur attentif peut se demander comment on peut inscrire un programme dans un dispositif à lecture seule comme la ROM. La réponse à cette question permet d'introduire les différentes sous-familles de ROM à savoir :

— les ROM pures,
— les PROM,
— les EPROM,
— les EEPROM.

Une ROM est programmée à sa fabrication. Il suffit alors que l'utilisateur fournisse au fabricant du composant ce qu'on appelle un masque qui définit parfaitement toutes les interconnexions microscopiques de la puce. Cette méthode ne convient bien évidemment qu'aux grandes séries industrielles et c'est probablement de cette manière qu'à été construite la ROM Sinclair.

Une PROM est une ROM programmable (programmable ROM) contenant une multitude d'interconnexions réalisées avec des fusibles microscopiques et c'est en provoquant des court-circuits aux bornes de broches bien déterminées qu'on fait fondre ces fusibles et de cette manière qu'on décide si telle case mémoire doit contenir un bit "1" ou "0". Ce type de programmation est irréversible et donc ne supporte pas les erreurs de programme.

Une EPROM est une ROM programmable et effaçable (ERASABLE AND PROGRAMMABLE ROM). La programmation se fait de manière électrique et l'effacement est obtenu en exposant le composant aux ultra-violets. Nous aurons l'occasion de revenir plus amplement sur ce type de mémoires car elles restent actuellement les mémoires mortes les plus utilisées par les amateurs.

Une EEPROM est une ROM programmable et effaçable électriquement (ELECTRICALLY ERASABLE AND PROGRAMMABLE ROM). La programmation comme l'effacement se fait de manière électrique ce qui est l'idéal pour l'électronique qui n'a alors plus besoin de recourir à des techniques annexes, mais ce composant reste assez nouveau et n'a pas encore une grande diffusion.

1.4. LA RAM

1.4.1. Définition-fonction

Elle constitue en quelque sorte le bloc-note du microprocesseur, c'est en RAM que s'implante le programme BASIC que l'utilisateur entre par le clavier.

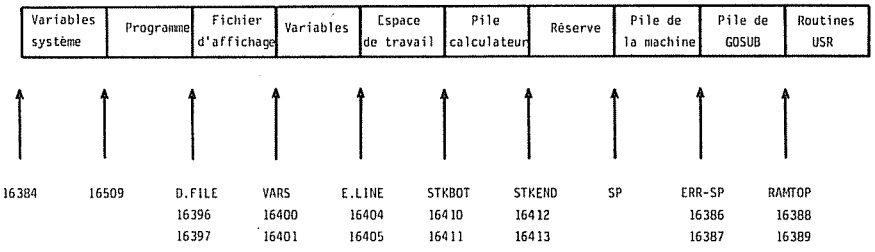
1.4.2. Dans le ZX 81

Dans le ZX 81 la RAM occupe les adresses de 4000H à 43FFH ou encore 16384 à 17403 en décimal, cela dans la version de base 1 K; pour la version 16 K la RAM occupe les adresses 4000H à 7FFFH soit 16384 à 32763 en décimal.

La segmentation de la zone RAM en plusieurs sous-zones est gérée directement par le logiciel système et il est bon d'en rappeler ici les grandes lignes:

La RAM est divisée en plusieurs blocs:

— RAM réservée aux variables sytèmes,
— zone programme,
— zone réservée à l'affichage sur écran,
— zone réservée aux données de l'utilisateur,
— zone réservée aux routines user éventuelles.



Segmentation de la RAM ZX 81.

La RAM est bien accessible en lecture et en écriture, le meilleur moyen de le vérifier est d'effectuer un PEEK et un POKE dans les adresses RAM et de s'apercevoir que ceci fonctionne :

Par exemple :

PRINT PEEK 17000 (lecture de la case mémoire RAM à l'adresse 17000)

On obtient une certaine valeur :

POKE 17000, 100 (écriture d'un 100 dans cette même case mémoire)
PRINT PEEK, 17000 (puis relecture)

On obtient alors la valeur 100 précédemment inscrite, et l'écriture dans un dispositif comme la RAM est effective.

Remarquons tout de même que cette manipulation ne fonctionne pas avec quelques variables systèmes en RAM, cela simplement parce qu'elles sont continuellement remise à jour par le BASIC.

1.4.3. Capacité-organisation

Une RAM comme une ROM est composée d'un très grand nombre de cellules mémoires. La capacité d'un tel dispositif est justement ce nombre de cellules élémentaires, elle est exprimée en kilobits ou kilo-octets.

1 kilobit = 2 ** 10 bits = 1 024 bits
1 kilo-octets = 1 024 octets = 8 192 bits

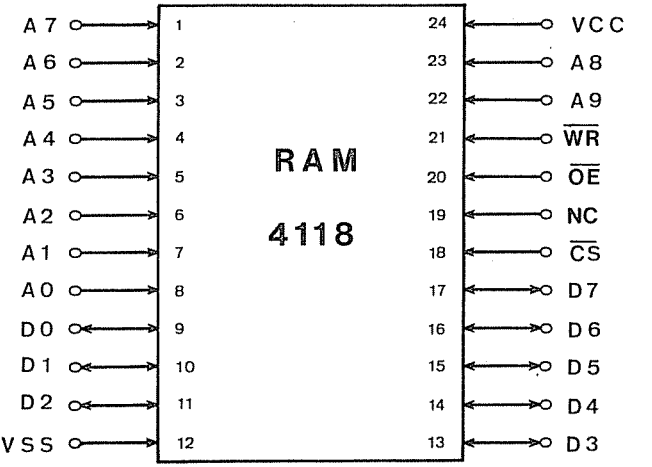
Avec la notion de capacité va aussi la notion d'organisation. Deux mémoires de même capacité peuvent avoir des organisations différentes : par exemple un ensemble de 8 192 cellules élémentaires de mémorisation de 1 bit peut être organisé :

soit avec 13 lignes adresses et une ligne de données → 8K * 1
soit avec 11 lignes adresses et 4 lignes de données → 2K * 4
soit avec 10 lignes adresses et 8 lignes de données → 1K * 8

ainsi si par exemple on désire constituer un bus de données de 8 lignes, il faudra avec les mémoires de type 1 mettre huit boîtiers identiques en

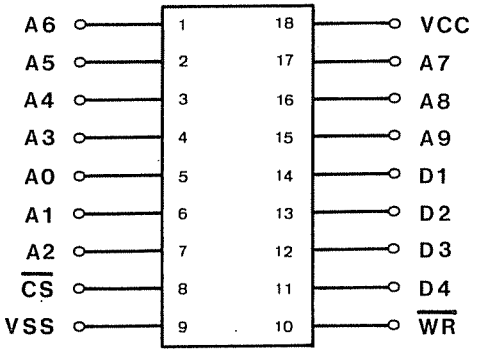
parallèle, avec les mémoires de type 2 mettre deux boîtiers identiques en parallèle et avec les mémoires de type 3 un seul boîtier suffira.

Dans le ZX 81 certaines versions de base sont câblées avec deux mémoires de type 2 114 dont la capacité est de 4 Kbits et l'organisation de 1 K * 4, d'autres sont câblées avec une mémoire de type 4 118 dont la capacité est de 1 Koctets et l'organisation de 1K * 8.



RAM 4118

2114



RAM 2114

1.4.4. Les broches

De manière classique on retrouve parmi les broches des RAM :

— une partie ou l'ensemble du bus de données,
— une partie ou l'ensemble du bus adresses,
— une entrée de sélection du boîtier : CS (CHIP SELECT) ou CE (CHIP ENABLE),
— les signaux de lecture et d'écriture : WR/ et RD/,
— les pattes d'alimentations.

1.4.5. RAM statique — RAM dynamique

Pour compléter l'information du lecteur sur les mémoires vives et aussi parce qu'il va les utiliser, disons encore qu'il existe deux sous-ensembles de RAM :

— Les RAM statiques : comme la 2114 et la 4118 qui s'interfacent facilement et directement avec le microprocesseur sans génération de signaux particuliers.

— Les RAM dynamiques : qui nécessitent un traitement particulier de la part du microprocesseur, mais qui en contre-partie sont plus rapides, plus intégrées, moins chères et consomment moins. Nous aurons l'occasion de revenir plus en détail sur ce type de mémoires vives.

1.5. LA PUCE SINCLAIR

Sinclair a conçu et réalisé un composant électronique sophistiqué et très spécifique au ZX 81.

Celui-ci se présente sous la forme d'un boîtier rectangulaire d'où sortent 40 broches, comme le microprocesseur. Il est repéré sur la photo de la carte ZX 81 par l'annotation "logique de calcul" et comporte l'inscription "Sinclair Computer Logic".

Ce composant a été imaginé pour intégrer sous un boîtier unique un ensemble de fonctions périphériques qui existait déjà dans le précurseur

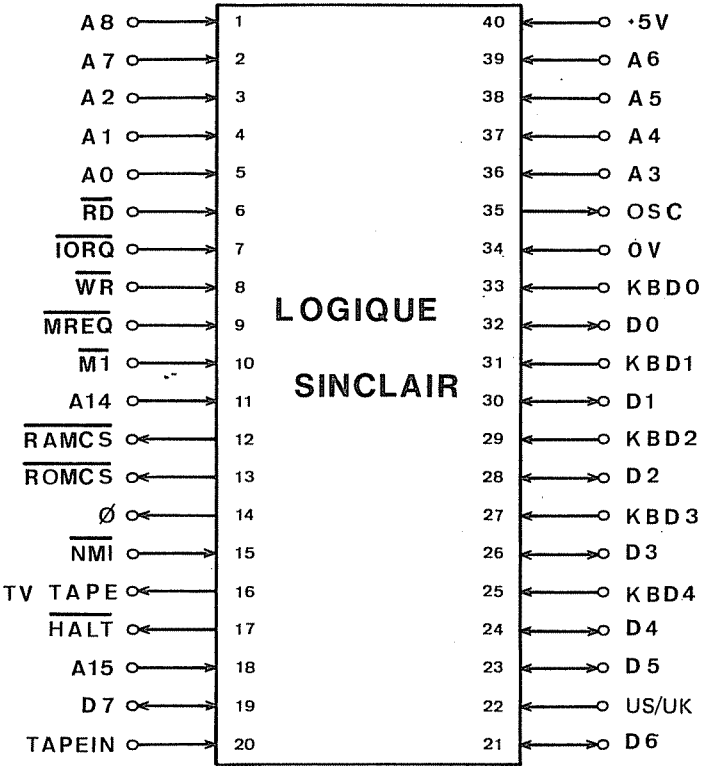
du ZX 81 : Le ZX 80, mais sous la forme de plusieurs portes logiques électroniques. A lui seul, il remplit la fonction de 18 boîtiers logiques standards.

J'attire quand même l'attention du lecteur sur le fait qu'un boîtier aussi spécialisé que celui-ci n'est pas approvisionnable par les fournisseurs de composants classiques, et que l'on doit donc prendre toutes les précautions pour éviter de le détériorer.

Les différentes fonctions qu'intègrent cette puce sont les suivantes :

— Il fait osciller le cristal de céramique, puis divise la fréquence du signal ainsi obtenu.

— Il s'occupe de sélectionner soit la ROM soit la RAM dans les différentes actions menées par le CPU.



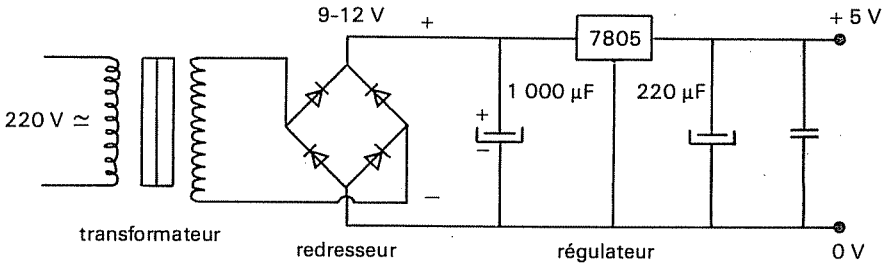
— Il prépare et met en forme le signal vidéo, en respectant le standard TV adéquat puis l'aiguille sur le modulateur TV.
— Il réalise l'interfaçage entre le système et le magnétophone à cassettes.
— Il réalise aussi la scrutation et le codage du clavier à touches sensitives.

Nous donnons le brochage de ce composant très particulier à cette application dans les lignes suivantes.

1.6. L'ALIMENTATION ZX 81

Quand on s'intéresse aux possibilités d'extension d'un système, il est bon de connaître parfaitement les caractéristiques de l'alimentation existante pour savoir si elle est suffisante, ou s'il faut en changer.

Les schémas de l'alimentation ZX 81 sont donnés ci-après :



Synoptique de l'alimentation ZX 81

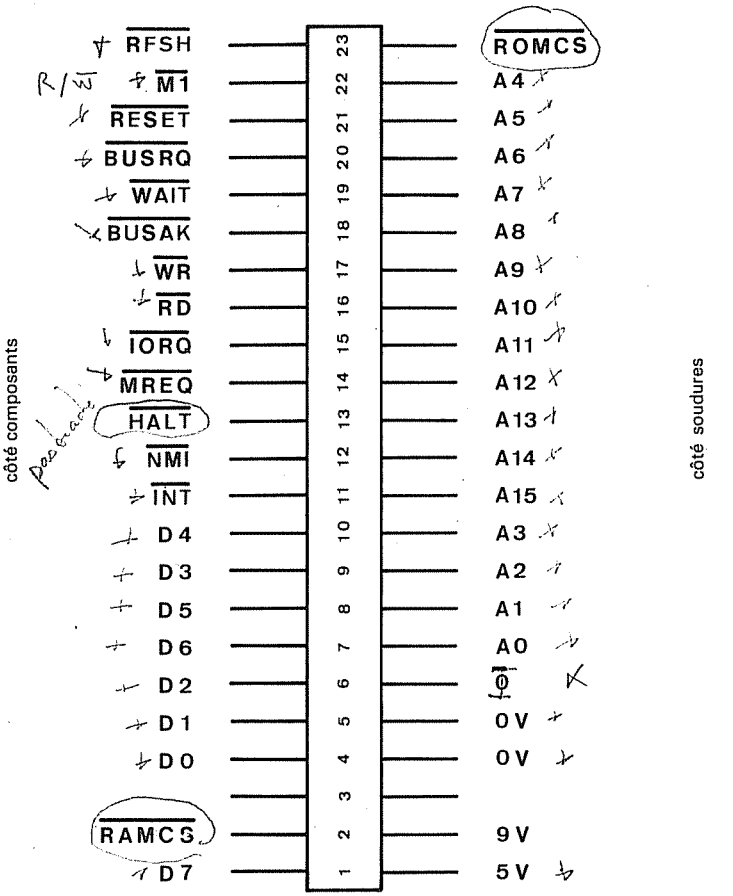
Son fonctionnement est parfaitement classique : on transforme la tension secteur en 9 V alternatifs qu'on redresse par un pont de graetz à base de diodes 1N 4 007, puis on filtre avec un condensateur chimique de 1 000 micro-farad.

La régulation en tension est effectuée dans le boîtier du micro-ordinateur lui-même à l'aide d'un composant intégré : le LM 7 805 monté sur un radiateur. C'est cette partie qui fait que votre ZX 81 diffuse de la chaleur.

Une telle alimentation peut fournir jusqu'à 1 ampère alors que la version de base consomme déjà 0,7 A, cela ne laisse que très peu de courant aux extensions éventuelles et nous serons dans l'obligation de revoir assez rapidement cette partie alimentation.

1.7. LE CONNECTEUR D'EXTENSION

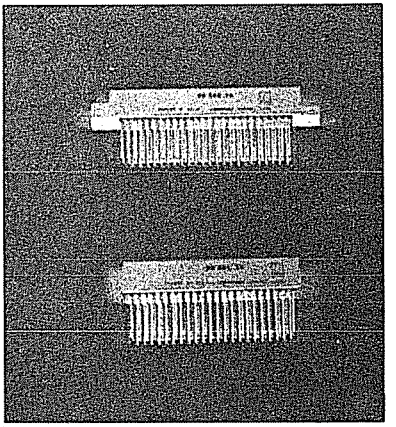
Vous avez remarquez à l'arrière de votre ZX 81, un connecteur mâle directement réalisé sur le circuit imprimé, réservé aux extensions (voir schéma).



Connecteur d'extension ZX 81.

Celui-ci comporte en tout 46 lignes soit 23 sur chaque face, il est au pas standard de 2,54 mm, seul son nombre de lignes n'est pas standard, ce qui oblige les lecteurs désireux de réaliser leurs propres extensions à confectionner un connecteur femelle eux-mêmes.

Pour cela il existe dans le commerce des connecteurs de ce type, encartable femelle, doublé face à souder ou à wrapper de deux fois 25 lignes que l'on peut transformer moyennant deux coups de scie bien appliqués en un deux fosi 23 lignes, utilisable pour le ZX 81 (voir photographie).

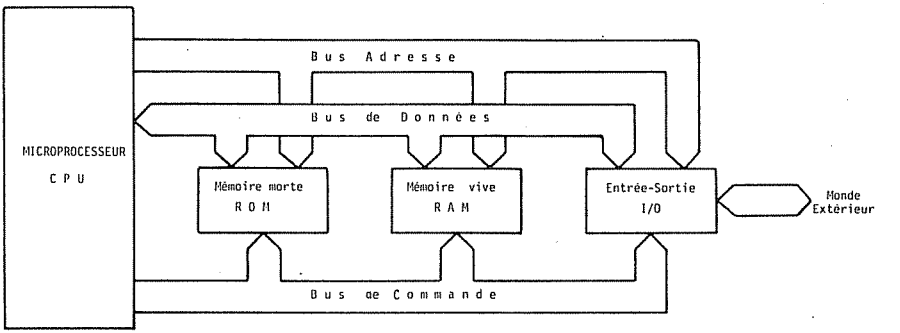


Réalisation du connecteur femelle.

Attention, pensez à faire des traits de scie obliques pour respecter exactement l'évasement vers le haut de la cavité du ZX 81 devant accueillir votre connecteur; ceci constituera un excellent détrompeur et assurera une bonne coïncidence entre les pistes cuivrées.

Une fois le connecteur réalisé, évitez tout de même de l'enficher lorsque le ZX 81 est sous tension, vous risquez de créer un court-circuit entre deux pistes, ce qui pourrait être fatal à quelques composants.

En guise de conclusion à ce chapitre de présentation du système ZX 81, nous vous invitons à regarder le schéma qui suit et dans lequel vous reconnaîtrez tous les composants qui constituent ce micro-ordinateur.



Architecture d'un système à microprocesseur.

2

L'outillage

Après la partie que nous pourrions qualifier d'encyclopédique et juste avant d'attaquer les réalisations pratiques proprement dites, il est opportun de dire quelques mots sur l'outillage et les techniques de câblage de l'électronicien. Cet aspect est loin d'être négligeable car un bricoleur si bon soit-il fait du très mauvais travail lorsqu'il a de mauvais outils.

Le premier outil de l'électronicien est bien évidemment le fer à souder; celui-ci doit faire entre 30 et 60 W pas plus car il risque alors de détériorer les composants en les chauffant excessivement, cela aussi parce que l'amateur encore novice aura tendance à garder son fer trop longtemps sur le composant et cela est tout à fait normal car cette pratique nécessite un certain coup de patte.

Pour faire de l'électronique deux techniques s'opposent avec chacunes leurs avantages: le circuit imprimé et le Wrapping.

2.1. LE CIRCUIT IMPRIMÉ

C'est la technique couramment employée dans l'industrie puisqu'elle se prête très bien à la réalisation des grandes séries. De plus elle a l'avantage de permettre des réalisations peut encombrantes et propres. Son principal inconvénient est qu'au niveau de l'amateur débutant et lorsqu'on travaille comme en micro-informatique avec des circuits imprimés très fins et parfois double-face, elle nécessite un investissement non négligeable.

Ses outils et son matériel spécifique sont:

— des pages de mylar et des pistes transfert pour dessiner le circuit imprimé,

— une table à insoler,

— des plaques epoxy cuivrées présensibilisées,

— une perceuse sur support ainsi que des forêts.

Nous donnerons parfois, dans la suite, des réalisations sur circuits imprimés pour les lecteurs que cela intéresse.

2.2. LA TECHNIQUE WRAPPING

Un peu moins bien connue du grand public et peut-être à cause de cela mal distribuée, elle consiste à réaliser des connexions en entourant autour d'une broche à section carrée (spécialement conçue à cet usage) un fil conducteur, relativement fin, et préalablement dénudé à son extrémité. Cette technique se prête bien à la micro-informatique, moins bien à une électronique qui nécessiterait plus de composants discrets ou qui mettrait en œuvre des puissances électriques plus grandes (le fil à wrapper ne peut pas compte tenu de sa section réduite supporter de grosses intensités). Le wrapping a néanmoins cet énorme avantage d'être très souple aux modifications de circuit puisqu'il est très aisé de déwrapper une connexion pour la changer de place.

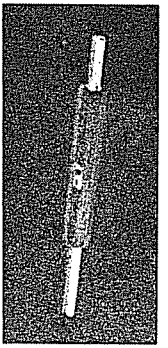
En wrapping, le support du circuit électronique n'est plus un circuit imprimé mais une carte à wrapper ; à cet usage, je conseillerai aux lecteurs d'utiliser une carte de test pour prototype percée au pas de 2,54 mm dans les deux dimensions (voir photo).

Il est important aussi, que les trous d'une telle carte soit isolés entre eux. La dimension retenue pour le montage dépendra du nombre d'extensions que désirera réaliser le lecteur ; mais sachez dès à présent qu'il ne faut pas voir trop petit et se retrouver finalement avec une carte trop courte pour continuer. Une carte de 100 mm * 200 mm permet d'implanter toutes les extensions proposées dans cet ouvrage, et nous proposons à l'annexe quatre un exemple de disposition.

La technique wrapping a une autre exigence, c'est l'utilisation systématique de supports pour tous les composants, supports à wrapper bien sûr. Ceux-ci existent dans les mêmes formats que les supports à souder et leurs prix s'équivalent ; leur seule différence est au niveau des pattes qui mesurent environ 25 mm dans la famille à wrapper (voir photo).

L'opération par elle-même s'effectue à l'aide d'un outil à wrapper et de fil à wrapper. L'outil à wrapper s'achète à un prix modique et il réalise à la fois le dénudage du fil, le wrapping et le dewrapping (*voir photo*).

Pour dénuder : on enfile le fil à wrapper sur une longueur de 3 cm environ dans la lame en forme de "V" située au milieu de l'outil, on appuie le fil au fond du "V" et on tire perpendiculairement à l'outil ; la gaine du fil est coupée puis expulsée.



Détail de l'outil à wrapper.

Pour wrapper: on enfile la partie dénudée du fil dans un tout petit trou aménagé en bout du côté à wrapper de l'outil, le fil à nu ne doit pas apparaître. Ensuite on place l'outil et le fil sur la broche à wrapper puis on tourne l'outil sans appuyer dessus et toujours dans le sens des aiguilles d'une montre jusqu'à ce qu'on ne sente plus aucune résistance. Le fil va s'entrelacer très proprement autour de la broche sur environ 5 spires.

Pour dewrapper: placer d'abord l'outil du côté dewrapping (le plus court) sur la broche que vous désirez dewrapper et tourner dans le sens contraire à celui des aiguilles d'une montre en appuyant légèrement. Vous allez sentir la connexion devenir lâche jusqu'à se défaire.

Encore quelques conseils avant de commencer réellement:

— Faites plusieurs essais.

— Si une connexion a ses spires mal juxtaposées, c'est que vous appuyez trop sur l'outil et il faut la recommencer sans hésiter.

— Un fil ne peut guère être wrappé puis dewrappé plus de deux fois sinon il risque de casser.

— On ne doit pas hésiter à recommencer une connexion mal isolée, c'est le cas lorsqu'on peut apercevoir à la base de la broche une partie de fil dénudé.

— On peut superposer jusqu'à trois enroulements wrapping sur la même broche.

Pour éviter que votre wrapping ne deviennent une vraie forêt de fils impénétrable, voici encore quelques conseils:

— Utilisez des fils à wrapper de couleurs différentes; par exemple jaune pour le bus des adresses, noir pour bus des données, vert pour le bus de contrôle etc... Cela vous aidera grandement dans le repérage des connexions.

— Attention lorsque vous wrappez, vous êtes du côté circuit et le numéro des broches est inversé par rapport aux schémas de brochage qui sont habituellement donnés en vus de dessus; par exemple: la broche numéro 1 est alors à droite et non plus à gauche.

— Câblez en premier lieu tout ce dont vous êtes certain, de façon à garder au-dessus ce qui est incertain, accessible pour d'éventuelles modifications.

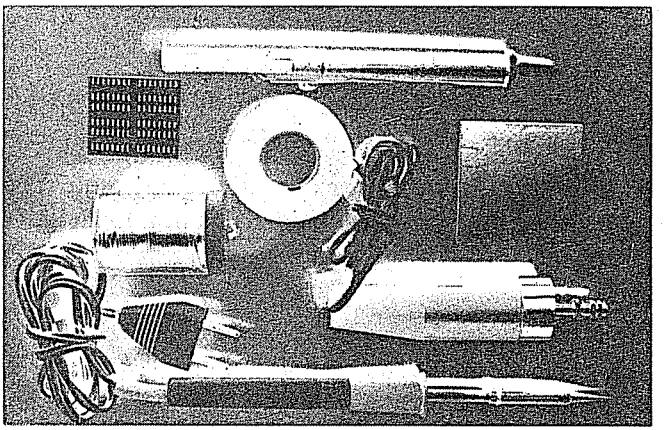
— On peut mettre un point de soudure sur les broches devant accueillir les alimentations, cela permet de maintenir le support plaqué contre la carte.

— Enfin il faut wrapper au plus court entre deux points et veiller à ce que les fils soient légèrement tendus, ce qui implique qu'il faut dénuder juste au bon endroit.

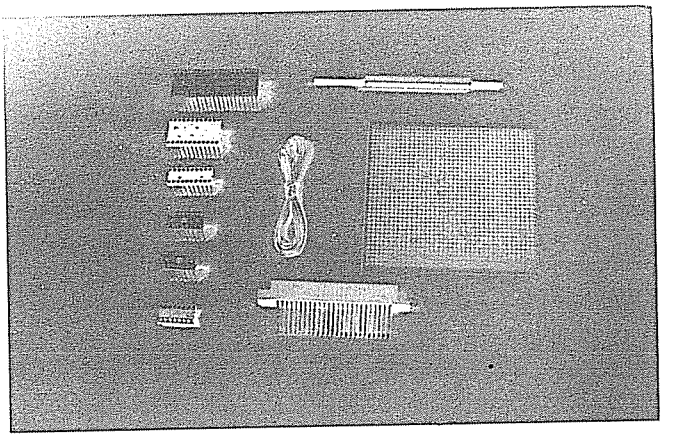
Nous récapitulons ici l'outillage élémentaire nécessaire au wrapping :

— carte test percée au pas 2,52 mm dans les deux dimensions,
— outil à wrapper,
— fil à wrapper (plusieurs couleurs de préférence),
— supports pour circuits intégrés à wrapper.

Cet ensemble est représenté sur la photo suivante.



L'outillage à souder.



L'outillage à wrapper.

Vous l'avez compris, c'est la technique du wrapping qui va être choisie pour les réalisations à venir. Mais que les partisans du circuit imprimé ne se désolent pas car il leur sera d'une très grande facilité de constituer eux-mêmes des masques, à partir des schémas électriques fournis pour chaque extension.

3
Le décodage d'adresses

3.1. INTRODUCTION

Avant d'entreprendre les extensions proprement dites, il faut préparer un terrain qui soit favorable, et plus particulièrement, générer des signaux de sélection supplémentaires, qui permettront de définir, à chaque instant, avec quel sous-ensemble du système, le CPU est en train de dialoguer ; cela se nomme le *décodage d'adresses*.

3.2. EXPOSÉ THÉORIQUE

3.2.1. Définition

De manière générale le décodage d'adresses consiste à imposer quelles vont être les adresses concernant la ROM, la RAM ou bien les périphériques, car on conçoit aisément que deux dispositifs dissemblables ne peuvent coexister à la même adresse, dans quel cas on parlera de conflit d'adresse.

Pour ce faire, on utilise les lignes adresses elles-mêmes, et en particulier les signaux de contrôle : MREQ/ et IORQ/, que l'on combine logiquement et astucieusement.

Mais reprenons d'abord plus en détail le rôle de ces deux derniers signaux de contrôle. Nous le rappelons, il s'agit là de deux sorties établies par le microprocesseur et destinées à sélectionner soit une mémoire, soit un périphérique de la manière suivante :

si MREQ/ = 0 → on sélectionne une mémoire
si IORQ/ = 0 → on sélectionne un périphérique

Pour accomplir cette sélection le CPU s'en remet entièrement à son programme où les instructions peuvent concerner le dialogue avec l'un ou l'autre des dispositifs nommés plus haut.

On donne à l'annexe 3 la liste des instructions travaillant sur les mémoires ainsi que la liste de celles qui mettent en cause uniquement des périphériques d'entrée ou de sortie.

Il faut savoir aussi qu'il n'est aucunement interdit d'adresser un périphérique à un emplacement normalement réservé à une mémoire, et c'est même parfois très utile.

3.2.2. Principe

C'est par combinaison logique des différents signaux évoqués plus haut (A0 – A15, MREQ/ et IORQ/) que l'on forme un nouveau signal composite, qui va venir activer les lignes de sélection de boîtier ou Chip Select (CS ou CS/) que nous avons rencontrées dans les descriptions des ROM et des RAM et qui existent aussi dans les périphériques d'entrée-sortie.

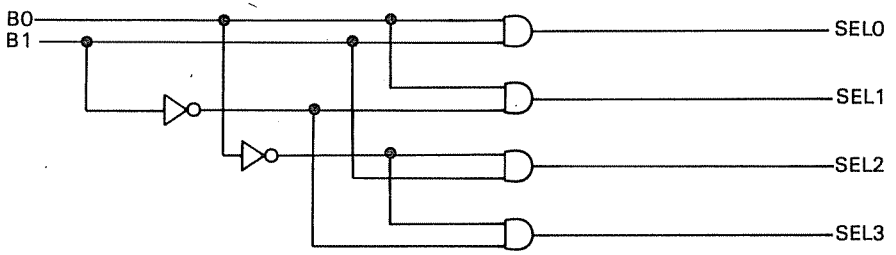
Par exemple avec deux lignes utilisables pour le décodage d'adresses on peut alors sélectionner quatre dispositifs différents :

En effet si on appelle B0 et B1 ces deux lignes on peut avoir :

soit B0 = 0 et B1 = 0
soit B0 = 1 et B1 = 0
soit B0 = 0 et B1 = 1
soit B0 = 1 et B1 = 1

et à chaque cas de figure on peut s'arranger pour générer par composition logique les signaux destinés à être relier au "Chip Select" des dispo-

sitifs. Un exemple d'une telle combinaison logique réalisée en utilisant seulement des portes simples est donné ci-dessous:



La capacité d'adressage du Z 80, c'est-à-dire le nombre maximal de positions mémoires ou de périphériques qu'il peut adresser est de 64 Koctets de mémoires car il peut alors utiliser ses 16 lignes d'adresses (2**16 = 64 * 1 024) et de 256 périphériques d'entrées-sorties car dans ce cas les huit lignes d'adresses de poids faibles et les huit lignes adresses de poids forts sont identiques et c'est donc seulement 8 des 16 lignes qui sont utilisables pour adresser les périphériques (2**8 = 256).

3.2.3. Sur le ZX 81

Nous savons donc que pour accéder à une position mémoire il faut connaître son adresse; cela sous-entend naturellement qu'à une adresse correspond au plus une position mémoire. L'inverse n'est pas forcément vrai et on peut parfois accéder à une mémoire donnée par plusieurs adresses différentes, c'est ce que l'on nomme un *adressage partiel.*

Le ZX 81 pour des raisons de simplification et d'économie de composants ne réalise qu'un adressage partiel, la meilleure vérification en est le programme BASIC suivant:

```
10 LET A = 16384
20 FOR I = 125 TO 145
30 PRINT I;
40 PRINT TAB  6; PEEK I;
50 PRINT TAB 12; PEEK (A+I);
60 PRINT TAB 18; PEEK (2*A+I);
70 PRINT TAB 24; PEEK (3*A+I)
80 NEXT I
```

et lorsque qu'on demande l'extension de ces lignes, on voit s'inscrire quatre colonnes de nombres :

— la première étant le contenu des cases mémoires dont les adresses sont comprises entre 125 et 145, donc normalement situées en ROM,

— la seconde étant le contenu des cases mémoires dont les adresses sont comprises entre 16384 + 125 et 16384 + 145, donc normalement situées en RAM et plus particulièrement dans la zone où est entré le programme BASIC,

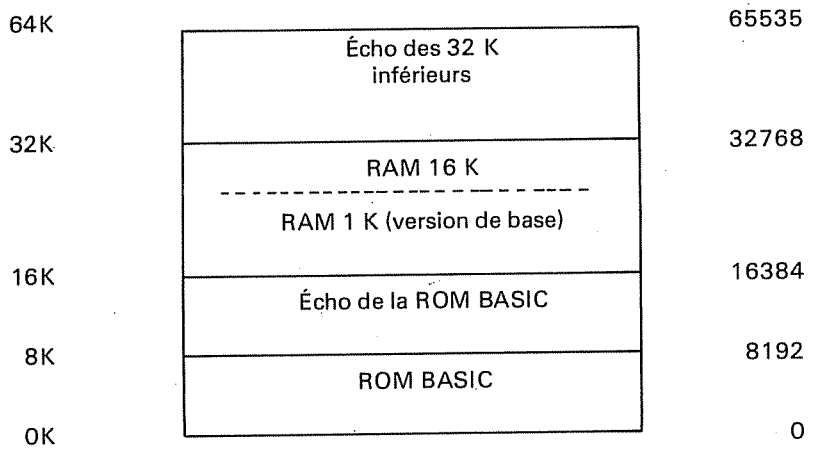
— la troisième étant le contenu des cases mémoires dont les adresses sont comprises entre 32768 + 125 et 32768 + 165,

— la quatrième étant le contenu des cases mémoires dont les adresses sont comprises entre 49152 + 125 et 49152 + 145.

On remarque que les colonnes 1 et 3 sont identiques ainsi que les colonnes 2 et 4, ce qui laisserait supposer que la mémoire commençant A 0, c'est-à-dire la ROM est la même que celle qui commence à 32768 (8000H) et de la même manière on aurait identité entre la mémoire en 16384 (4000H), c'est-à-dire la RAM et la mémoire en 49152 (C000H).

En investiguant encore dans la carte mémoire (MEMORY MAP) du ZX 81 on obtient finalement les résultats consignés dans le diagramme ci-dessous.

Carte mémoire du ZX 81



Tout cela s'explique si on admet que seul le bit A14 du bus adresses est utilisé pour décoder les mémoires. Autrement dit:

— Si A14 = 0, on adresse la ROM, c'est par exemple le cas dans les adresses:

```
DE  0000 0000 0000 0000  B = 0000H
A   0011 1111 1111 1111  B = 3FFFH

ET  1000 0000 0000 0000  B = 8000H
A   1011 1111 1111 1111  B = BFFFH
```

— Si A14 = 1, on adresse la RAM, c'est par exemple le cas dans les adresses:

```
DE  0100 0000 0000 0000  B = 4000H
A   0111 1111 1111 1111  B = 7FFFH

ET  1100 0000 0000 0000  B = C000H
A   1111 1111 1111 1111  B = FFFFH
```

D'où un énorme gaspillage de positions mémoires dans la version de base:

— 32 Koctets de mémoires pour la ROM qui ne nécessite qu'en fait 8K,

— 32 Koctets de mémoires pour la RAM qui ne nécessite qu'en fait 1K.

3.2.4. Décodage plus complet

Pour obtenir un décodage plus complet, nous allons proposer un petit circuit logique. Mais au préalable, voyons sur l'exemple du ZX 81, comment se traite un véritable problème de décodage d'adresses.

La logique binaire va nous être d'un grand secours alors j'espère que vous n'avez pas tout oublié sinon GOTO annexe 1.

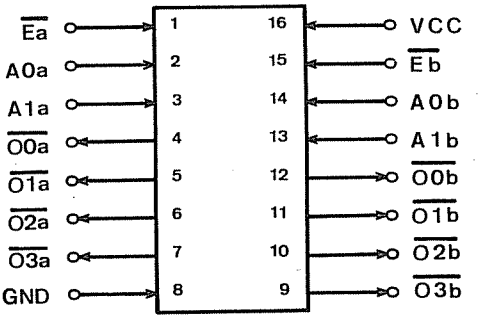
Si nous considérons le ZX 81, en version 16 K, le problème consiste à adresser la ROM 8K et la RAM 16K tout en se laissant la possibilité d'exploiter les autres zones d'adressage par les extensions à venir.

La taille de la plus grande zone mémoire, ici la RAM avec 16K, permet d'envisager un premier morcelage de la capacité d'adressage du microprocesseur en quatre pages de 16 Koctets. Cela est obtenu en utilisant les deux lignes adresses les plus hautes A14 et A15, de la manière suivante:

A14	A15	
0	0	page de 0 à 16K
1	0	page de 16 à 32K
0	1	page de 32 à 48K
1	1	page de 48 à 64K

avec de plus MREQ/ = 0

En électronique il existe un composant qui réalise une telle fonction, il s'agit d'un décodeur quatre voies, et pour fixer les idées on peut par exemple utiliser un 74LS139 qui contient deux décodeurs quatre voies (voir brochage et table de vérité ci-contre).



Double décodeur 2-4: 74 LS 139

Table de vérité du décodeur quatre voies

A1	A0	E	00/	01/	02/	03/
0	0	0	0	1	1	1
0	1	0	1	0	1	1
1	0	0	1	1	0	1
1	1	0	1	1	1	0
X	X	1	1	1	1	1

X état indifférent

Son montage est relativement aisé: il suffit de monter les deux lignes utilisées dans la sélection (A14 et A15) sur les lignes A0A et A1A, de câbler correctement les broches d'alimentation et de sélecter le décodeur lui-même en appliquant le signal MREQ/ sur son entrée EA/.

Les sorties O0A/, O1A/, O2A/ et O3A/ sont ensuite directement reliées aux CS/ des mémoires.

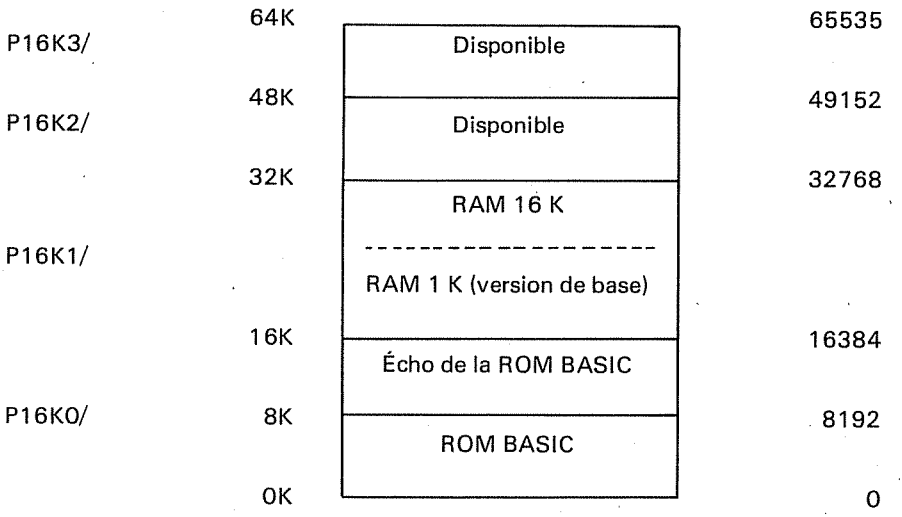
On nommera dans la suite les différents signaux issus de ces broches de la manière suivante:

pour O0A/ on aura P16K0/ pour page de 16 Koctets numéro 0
pour O1A/ on aura P16K1/ pour page de 16 Koctets numéro 1
pour O2A/ on aura P16K2/ pour page de 16 Koctets numéro 2
pour O3A/ on aura P16K3/ pour page de 16 Koctets numéro 3

Le schéma de la page suivante explicite encore d'avantage toute ces connexions à effectuer.

La carte mémoire d'origine est alors modifiée et on obtient:

Carte mémoire du ZX 81 après modification 1



N.B.: la colonne tout à fait à gauche indique le signal de sélection utilisé.

3.2.5. Décodage encore plus complet

Mais lorsqu'on explore la première page ainsi définie, on s'aperçoit que la ROM n'a effectivement besoin que de 8 koctets sur les 16 qui lui sont attribués, d'où l'idée de décoder plus encore cette zone mémoire comprise entre 0 et 16 K.

Il faut donc, comme nous l'avons vu et en effectuant des combinaisons logiques entre les signaux de manière astucieuse, différencier les deux zones : 0 à 8K et 8 à 16K

La première page de 8 Koctets s'étend des adresses :

DE 0000 0000 0000 0000B = 0000H
A 0001 1111 1111 1111B = 1FFFH

et la seconde :

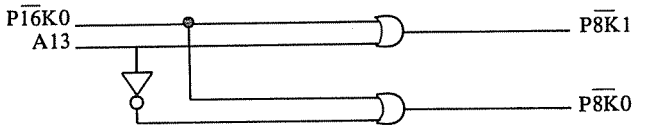
DE 0010 0000 0000 0000B = 2000H
A 0011 1111 1111 1111B = 3FFFH

Nous constatons que toutes les adresses de la page de 0 à 8K ont en commun leur bit A13 à "0" alors que toutes les adresses de la page de 8 à 16K ont leur bit A13 à "1"; les bits A14 et A15 étant dans les deux cas égaux à zéro.

Cela suffit pour décoder ces deux sections. Nous allons utiliser le signal P16K0/ réalisé plus haut et qui équivaut en fait à : P16K0/ = A14/.A15/, pour le combiner avec le signal A13 dans les relations qui suivent et afin de former deux nouveaux signaux de sélection des pages de 8 Koctets :

P8K0/ = P16K0/ . A13/
P8K1/ = P16K0/ . A13

ce qui donne aussi avec des portes logiques :



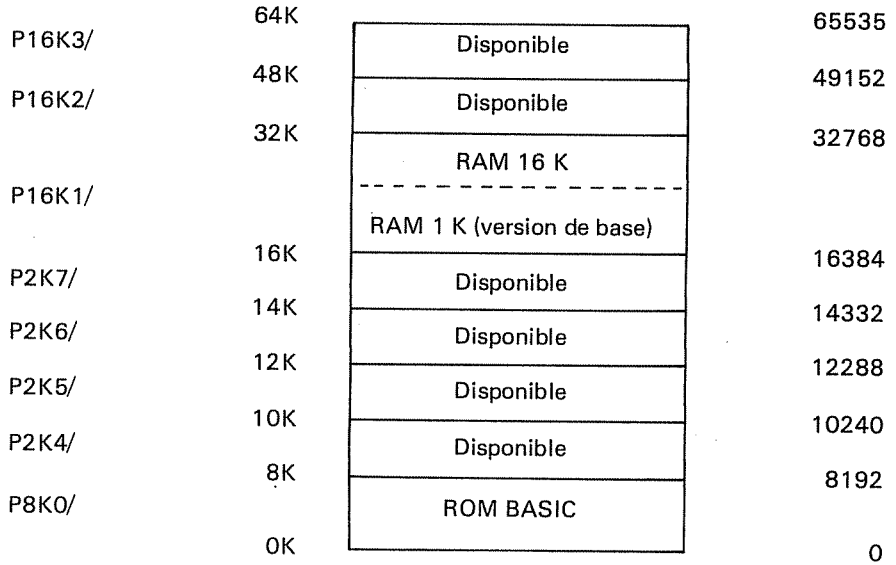
Nous pourrions réserver à la section de 8K située entre les adresses de 8 à 16K une utilisation tout à fait particulière qui est de servir à adresser des EPROM qui pourraient venir en addition sur le système. Des EPROM faciles à approvisionner par l'amateur et ayant un coût réduit sont les 2716, produit courant chez les constructeurs. Elles ont une capacité de 2 Koctets et donc on peut en mettre jusqu'à quatre dans la zone isolée. Un nouveau problème de décodage d'adresse apparaît aussitôt et nous pouvons le résoudre aussi vite. Pour cela il suffit d'utiliser le second décodeur 1—4 laissé libre dans notre 74LS139 de la même manière que précédemment, en utilisant comme signal de sélection du décodeur le signal P8K1/ cité plus haut et les lignes adresses A11 et A12 pour effectuer le décodage proprement dit.

On synthétise par ce procédé sur les sorties du décodeur les signaux logiques de sélection suivants :

sur O0B/ on aura P2K4/ pour page de 2 koctets numéro 4
sur O1B/ on aura P2K5/ pour page de 2 koctets numéro 5
sur O2B/ on aura P2K6/ pour page de 2 koctets numéro 6
sur O3B/ on aura P2K7/ pour page de 2 koctets numéro 7

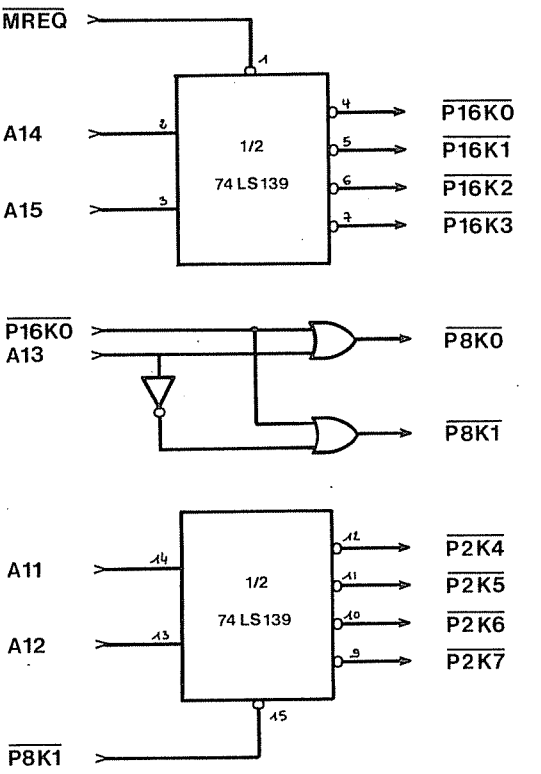
Cette seconde modification fait alors évoluer la carte mémoire de la manière suivante :

Carte mémoire du ZX81 après modification 2

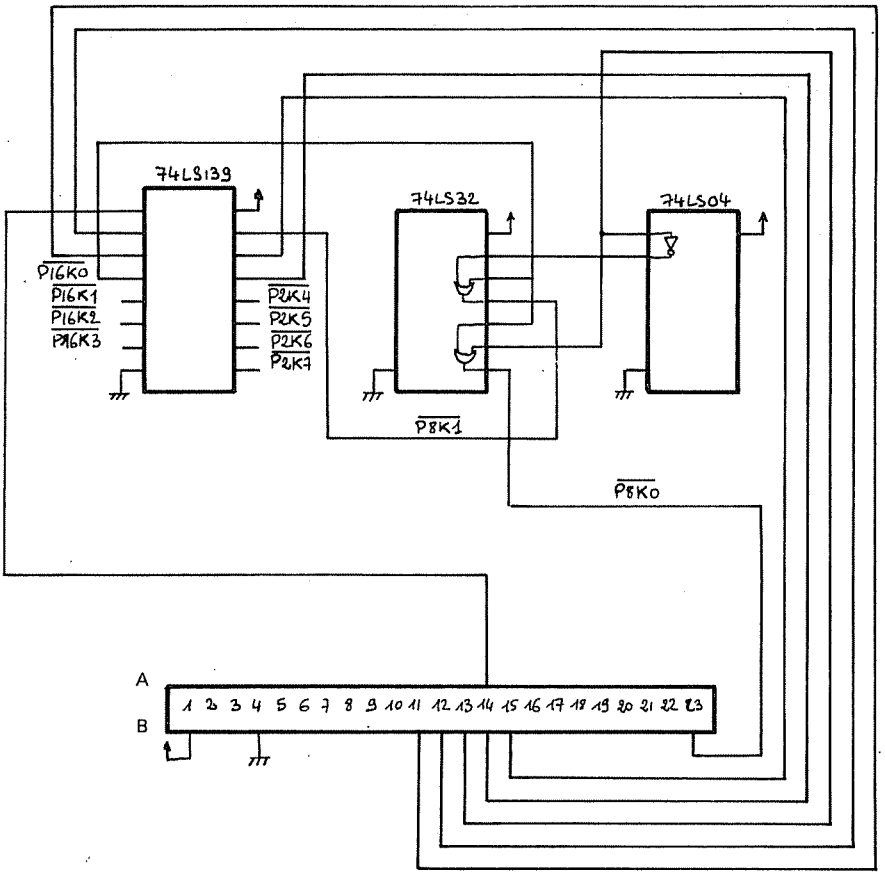


Il est bien entendu que ces nouveaux signaux de sélection pourront aussi être utilisés pour adresser des composants autres que des EPROM.

Nous donnons un schéma définitif de toute cette partie de décodage ainsi qu'une liste des composants utiles dans les pages qui suivent.



Décodage d'adresses (synoptique).



Décodage d'adresses (schéma de câblage).

3.3. RÉALISATION PRATIQUE

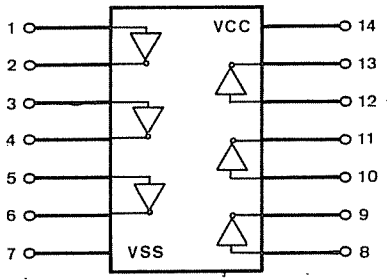
3.3.1. Instructions de montage

Voilà, nous attaquons notre première réalisation d'extension par ce petit circuit facile à faire fonctionner, mais au préalable il faut préparer la carte destinée à recevoir notre travail. Pour cela il suffit d'enficher les broches à wrapper du connecteur préparé comme nous l'avons décrit au

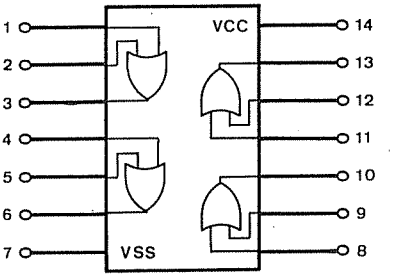
chapitre 1, au bord de la carte de test. Pour fixer l'ensemble on peut déposer une goutte de soudure à chacun des coins. Un exemple d'implantation est délivré à l'annexe 4.

Sur la carte de test équipée du connecteur femelle comme nous l'avons vu plus haut on ajoute trois supports à wrapper : un 16 broches destiné au 74LS139 et deux 14 broches pour le hexuple inverseur 74LS04 et la quadruple porte OR. On pourra éventuellement suivre la disposition préconisée dans l'annexe 4.

Comme toujours on commence à mettre un point de soudure sur les broches qui doivent recevoir les alimentations puis on commence à wrapper en suivant le schéma vu plus haut.



Hexuple inverseur 74LS04



Quadruple porte OR: 74LS32

Avant d'insérer les circuits intégrés effectuez un essai à vide : c'est-à-dire le ZX 81 avec sa carte d'extension privée de ses composants, simplement pour vérifier que la carte d'extension n'introduit pas de court-circuit dans le système. Dans quel cas il vous faudrait revoir votre travail de câblage.

La mise en fonctionnement définitif de la logique de décodage nécessite une petite intervention sur le ZX 81, il s'agit d'aller dessouder la résistance R 28 de 680 OHMS placée en série sur la ligne ROMCS/ entre le boîtier développé par Sinclair et la RAM interne. Cette opération n'est pas très compliquée, il suffit d'un peu de soin. La première recommandation est de ne pas laisser le ZX 81 sous-tension pendant l'intervention, la seconde d'éviter d'approcher trop le fer à souder des boîtiers du ZX 81.

La justification de cette ablation est d'enlever le contrôle du ROMCS/ au boîtier Sinclair pour pouvoir en générer un autre plus optimisé extérieurement.

Le système étendu est ensuite remonté et complété en insérant les circuits intégrés sur leur support correspondant. A la mise sous-tension votre ZX 81 apparaîtra inchangé : un écran vide avec curseur K ; si maintenant on lance à nouveau le programme BASIC précédent, on s'apercevra que les colonnes 1 et 3 sont devenues tout à fait dissemblables : votre ZX 81 peut maintenant utiliser les mémoires de 32 à 48 K et celle de 8 à 16 K pour des extensions, il dispose de plus pour cela de sept lignes de sélection supplémentaires.

3.3.2. Nomenclature des composants utilisés

Nous donnons sous la forme d'un tableau la nomenclature complète des composants utilisés dans cette extension :

Nombre	*Nomenclature*	*Description*	*Commentaire*
1	74LS139	Double décodeur 2→4	Exiger du LS
1	74LS04	Hexuple inverseur	Exiger du LS
1	74LS32	Quadruple porte OR	Exiger du LS
2	Support 14 broches	A wrapper	
1	Support 16 broches	A wrapper	
1	Carte de test	Dimensions : 100*200 mm	A pastilles isolées
1	Connecteur femelle encartable, 25 broches, double face		Chaque adjectif compte. 25 broches ou plus éventuellement

4

Interfaçage d'un coupleur parallèle

4.1. PRÉSENTATION

Une fois le décodage d'adresses correctement accompli, nous sommes maintenant en mesure de nous attaquer à notre première extension.

Lors du lancement publicitaire de son micro-ordinateur, Sinclair a utilisé comme argument que le ZX 81 serait capable de piloter une centrale nucléaire. Cela doit être pris de manière très restrictive car dans sa configuration de base le micro-ordinateur n'a pratiquement aucun moyen à sa disposition pour converser avec le monde extérieur ; nous faisons bien sûr abstraction des liaisons avec les dispositifs comme l'écran, le magnétophone ou le clavier, qui sont des liaisons normalement utilisées par le système et peu accessibles à l'utilisateur.

Un premier pas permettant de comprendre comment à partir d'un petit système comme le ZX 81 et moyennant quelques petites extensions, on pourrait à la rigueur piloter un ensemble aussi gigantesque qu'une centrale nucléaire est d'appréhender plus en détail la notion de périphérique d'entrées-sorties. On nomme aussi ces dispositifs des *coupleurs* et il existe actuellement deux grandes familles de coupleurs :

— les coupleurs parallèles qui transportent les informations binaires sur huit lignes parallèles et simultanément. On constitue ainsi un nouveau bus qui peut être utilisé dans un sens comme dans l'autre,

— les coupleurs série qui n'utilisent qu'une seule ligne de transmission et donc qui sont obligés avant toute opération de transmission d'effectuer une mise en série des données (sérialisation) et après toutes réceptions de réaliser l'opération inverse, c'est-à-dire une transformation série-parallèle ou désérialisation.

Le coupleur que nous désirons utiliser dans une première manipulation est du type coupleur parallèle. Les applications d'un tel dispositif sont innombrables et on peut donner parmi les exemples d'utilisations les plus courantes:

— le décodage d'un clavier organisé sous forme matricielle,

— la liaison de certaines imprimantes avec un système à microprocesseur,

— la réalisation de programmateurs d'EPROM,

— la liaison dans un système multi-processeurs entre les différents processeurs,

— le couplage de convertisseurs analogiques-digitaux ou digitaux-analogiques,

— le pilotage de moteurs pas à pas,

— la réalisation d'automates programmables,

— etc...

4.2. EXPOSÉ THÉORIQUE

4.2.1. Description-brochage

Le coupleur parallèle que nous avons en vue est un 6821 composant spécialisé fabriqué par Motorola aux États-Unis et EFCIS en France. On le nomme aussi PIA pour Peripheral Adapter Interface ou Adaptateur pour Interface Parallèle. Celui-ci est assez facile à approvisionner chez un fournisseur de composants et il a le grand mérite d'avoir un prix des plus compétitifs pour sa spécialité.

Extérieurement cette puce se présente dans le même format qu'un microprocesseur c'est-à-dire, un boîtier standard avec quarante broches.

Parmi les broches on retrouve :

— le bus de données (8 bits) bidirectionnel : D0 à D7,

— le RESET/ pour la remise à l'état initial du PIA en même temps que celle du microprocesseur,

— une entrée R/$\overline{W}$ pour indiquer si l'accès se fait en lecture ou en écriture,

— une entrée de validation : E (ENABLE) généralement connectée à l'horloge du système et permettant de synchroniser les actions du PIA sur celle du CPU,

— trois entrées de sélection du boîtier (CHIP SELECT) : CS0, CS1 et CS2/,

— deux entrées de sélection des registres internes du PIA : RS0 et RS1,

— deux sorties de reconnaissance d'interruptions (INTERRUPT REQUEST) : IRQ0 et IRQ1,

— deux bus périphériques de huit bits, bidirectionnels : PA0 – PA7 et BP0 – PB7 on nomme aussi ceux-ci communément des *ports* ; ports d'entrée lorsqu'ils reçoivent des données et ports de sortie lorsqu'ils en émettent,

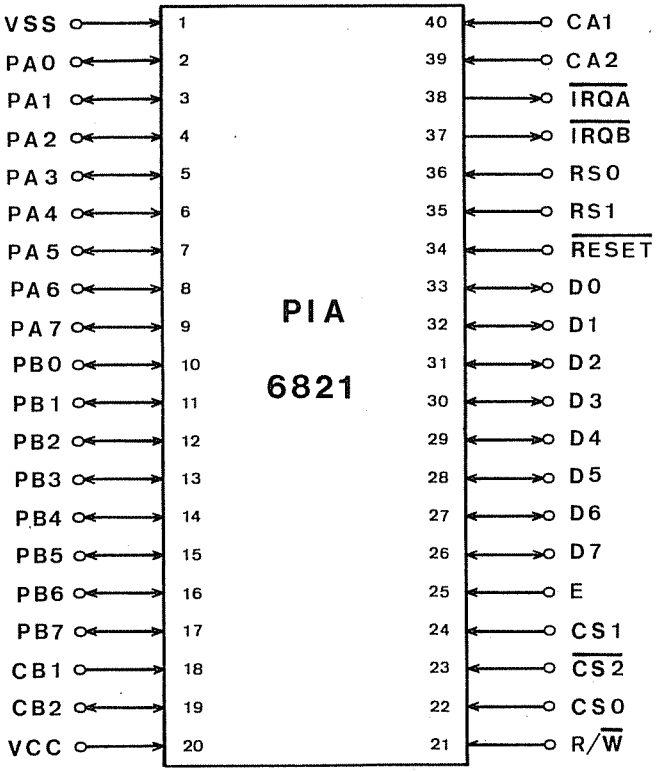
— quatre lignes de contrôle d'interruptions : CA1, CA2, CB1 et CB2,

— deux pattes d'alimentation : VCC et VSS.

Le schéma ci-après figure les broches du PIA et leur direction.

4.2.2. Architecture interne

L'organisation interne est relativement complexe du fait que ce composant est au même titre qu'un microprocesseur le résultat d'une intégration d'un grand nombre de fonctions logiques élémentaires. Nous ne l'exposerons donc pas en détail, mais ce qu'il est important de savoir est qu'un PIA comporte six registres internes de huit bits, trois pour le port A et trois pour le port B.



PIA 6821

Ainsi nous trouvons :

— le registre de données du port A noté ORA,
— le registre de données du port B noté ORB,
— le registre de direction des données du port A noté DDRA,
— le registre de direction des données du port B noté DDRB,
— le registre de contrôle du port A noté CRA,
— le registre de contrôle du port B noté CRB.

Ces six registres sont accessibles en sélectionnant correctement les lignes RS0 et RS1 vues plus haut sur le PIA. Un tableau permet de comprendre assez rapidement de quelle manière cela est réalisé :

Registre sélectionné	RS1	RS0
ORA	0	0
DDRA	0	0
CRA	0	1
ORB	1	0
DDRB	1	0
CRB	1	1

Une chose peut vous paraître surprenante, c'est de trouver aux mêmes adresses les registres de données (ORA ou ORB) et les registres de directions des données (DDRA ouDDRB) ; l'explication en est que la différentiation entre ces deux types de registres se fait par programmation et c'est ce que nous allons voir maintenant.

4.2.3. Programmation

Un PIA est un périphérique programmable, ce qui signifie que le CPU peut lui envoyer des ordres via son bus de données. Ces ordres peuvent concerner plusieurs opérations comme par exemple :

— choisir la direction des lignes du périphérique,
— générer une impulsion d'interruption,
— valider les interruptions...

La programmation est assurée par l'intermédiaire du registre de contrôle. La constitution de celui-ci n'est pas très simple, mais il est nécessaire de la voir en détail afin d'être en mesure d'utiliser plus tard pleinement les performances du PIA, même si les applications proposées dans la suite de ce livre ne les mettent pas toutes en œuvre.

Le registre de contrôle, pour un port donné, répond à un format tout à fait particulier ou chacun des huit bits qui le compose a une fonction précise. Ce format est le suivant :

Bit	B7	B6	B5 B4 B3	B2	B1 B0
CRA	IRQA1	IRQA2	Commande de CA2	Accès à DDRA	Commande de CA1
CRB	IRQB1	IRQB2	Commande de CB2	Accès à DDRB	Commande de CB1

Adoptons tout de suite la notation suivante qui va nous servir par la suite; nous appellerons CRA-I le I-ème bit du registre CRA (idem pour CRB).

Un tel format nécessite quelques explications, alors reprenons chaque bit:

A) CRA-2 et CRB-2

Le bit 2 de chaque registre de contrôle permet d'accéder soit au registre de données lorsque celui-ci est à zéro, soit au registre de directions des données lorsqu'il est à un. La table suivante permet de clarifier tout cela.

CRA-2	CRB-2	Registre sélecté
0	X	ORA
1	X	DDRA
X	0	ORB
X	1	DDRB

X = niveau indifférent

B) CRA-0, CRA-1 (RESP CRB-0, CRB-1)

Les deux bits de poids faible des registres de contrôle sont utilisés pour commander les entrées d'interruption CA1 et CB1. On peut en programmant ces bits choisir le front actif de CA1 (RESP CB1) et décider si une interruption reçue par CA1 (RESP CB1) doit être renvoyée en écho sur la broche de reconnaissance d'interruption IRQA (RESP IRQB) ou non.

De cette manière le bit CRA-0 (RESP CRB-0) est utilisé pour autoriser la sortie du signal de demande d'interruption IRQA (RESP IRQB):

si CRA-0 (RESP CRB) = 0 une interruption survenant sur CA1 (RESP CB1) est ignorée par la sortie IRQA (RESP IRQB) qui reste à l'état haut
si CRA-0 (RESP CRB-0) = 1 une interruption survenant sur CA1 (RESP CB1) est prise en compte par la sortie IRQA (RESP IRQB) qui passe à l'état bas

Le bit CRA-1 (RESP CRB-1) permet de choisir la transition active du signal d'entrée d'interruption CA1 (RESP CB1) ; celle-ci pouvant être soit un front montant c'est-à-dire une transition d'un niveau 0 à un niveau 1, soit un front descendant c'est-à-dire une transition d'un niveau 1 à un niveau 0.

si CRA-1(RESP CRB-1) = 0 on choisit de réagir sur un front descendant
si CRA-1 (RESP CRB-1) = 1 on choisit de réagir sur un front montant

Le tableau suivant rassemble ces résultats :

CRA-1 (CRB-1)	CRA-0 (CRB-0)	Front actif de CA1 (CB1)	Reconnaissance d'interruption IRQA (IRQB)
0	0	Descendant	Non
0	1	Descendant	Oui
1	0	Montant	Non
1	1	Montant	Oui

C) CRA-6, CRA-7 (RESP CRB-6, CRB-7)

Les deux bits de poids forts des registres de contrôle servent d'indicateur d'interruptions, cela signifie qu'ils mémorisent les interruptions qui ont pu survenir sur les broches CA1, CB1, CA2 et CB2.

Un zéro dans l'un quelconque de ces bits indique qu'aucune interruption n'est survenue, alors qu'un 1 permet de détecter l'intervention d'une interruption.

CRA-7 (RESP CRB-7) est attribué à la broche CA1 (RESP CB1)
CRA-6 (RESP CRB-6) est attribué à la broche CA2 (RESP CB2) lorsque celle-ci est en entrée

Pour faire retomber ces bits à zéro, après une interruption, il suffit de faire lire par le microprocesseur les registres de données correspondants : ORA pour resetter CRA-7 et CRA-6 et ORB pour resetter CRB-7 et CRB-6.

D) CRA-3, CRA-4,CRA-5 (RESP CRB-3, CRB-4, CRB-5)

Accrochez-vous bien car voici l'ensemble le plus compliqué. Ces trois bits commandent les lignes d'interruption CA2 et CB2. Le bit CRA-5 (RESP CRB-5) permet de déterminer si la ligne de contrôle d'interruption est utilisée comme entrée d'interruption ou comme sortie de commande.

Si CRA-5 (RESP CRB-5) = 0, CA2 (RESP CB2) est une interruption similaire à CA1 (RESP CB1) et sa programmation peut être expliquée de la même manière dans un tableau :

CRA-5 (CRB-5)	CRA-4 (CRB-4)	CRA-3 (CRB-3)	Front actif de CA1 (CB1)	Émission d'interruption IRQA (IRQB)
0	0	0	Descendant	Non
0	0	1	Descendant	Oui
0	1	0	Montant	Non
0	1	1	Montant	Oui

Si CRA-5 (RESP CRB-5) = 1, CA2 (RESP CA1) est une sortie de commande de la périphérie pouvant être utilisée comme système de synchronisation des échanges.

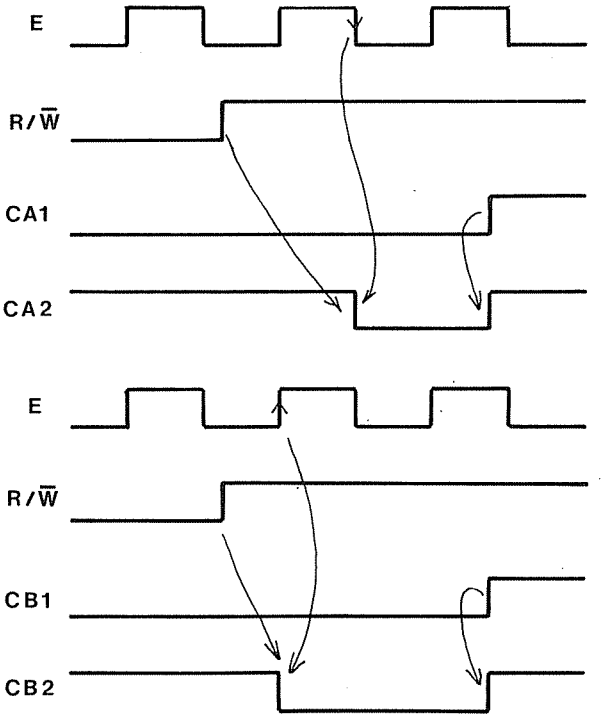
— Si CRA-4 (RESP CRB-4) = 1, la sortie CA2 (RESP CB2) suit le bit CRA-3 (RESP CRB-3) dont la valeur est fixée par les opérations d'écriture du CPU dans CRA (RESP CRB), autrement dit CA2 (RESP CB2) est à l'état bas quand CRA-3 (RESP CRB-3) = 0 et inversement CA2 (RESP CB2) est à l'état haut quand CRA-3 (RESP CRB-3) = 1.

— Si CRA-4 (RESP CRB-4) = 0, il y a deux types de fonctionnements : CA2 et CB2 ont alors des caractéristiques légèrement différentes :

— Si CRA-3 = 0, CA2 est mis à 0 par la transition négative du signal ENABLE suivant immédiatement une opération de lecture de ORA. CA2 est remis à 1 par un signal CA1 issu du périphérique ; ceci a pour but d'avertir le périphérique qu'il a été lu et permet de répondre par CA1 ; c'est ce que l'on nomme du "Handshaking".

Si CRB-3 = 0, l'opération du côté B est menée de la même façon à ceci près que CB2 est maintenu à zéro pendant la durée de l'écriture dans ORB puis remis à "1" par le signal CB1 suivant.

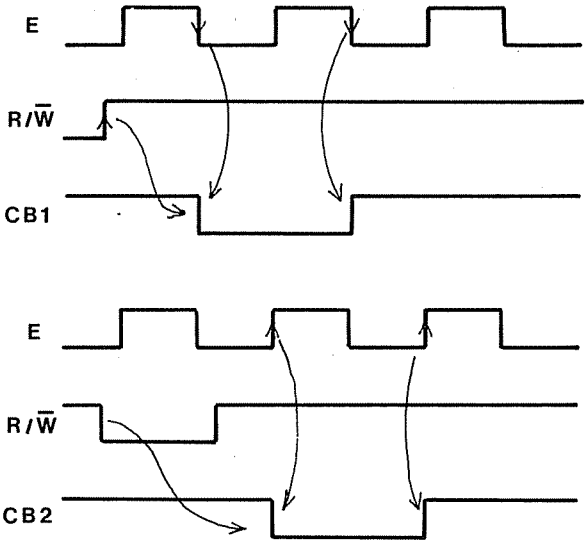
— Si CRA-3 = 1, CA2 est mis à 0 par la transition négative du signal ENABLE suivant immédiatement une opération de lecture de ORA, mais il est remis à 1 par la transition négative suivante du signal ENABLE.
Si CRB-3 = 1, c'est la même chose du côté B sauf que CB2 est mis à 0 par une transition négative de ENABLE suivant une opération d'écriture dans ORB. On parle de fonctionnement "PULSE STROBED".



Mode de fonctionnement PIA : Handshaking.

Nous donnons dans la suite les timings correspondants à la description de ces fonctionnements.

Cette partie étant assez difficile à saisir, il nous à sembler utile de rassembler toute ces informations sur la programmation d'un PIA dans une planche qui est donnée dans les pages suivantes.



Mode de fonctionnement PIA : Pulse Strobed.

4.2.4. Initialisation

Forts que nous sommes de tout cet enseignement, nous pouvons maintenant voir comment cela peut être utilisé.

Lorsqu'on applique un RESET sur un PIA celui-ci se réinitialise, c'est-à-dire :

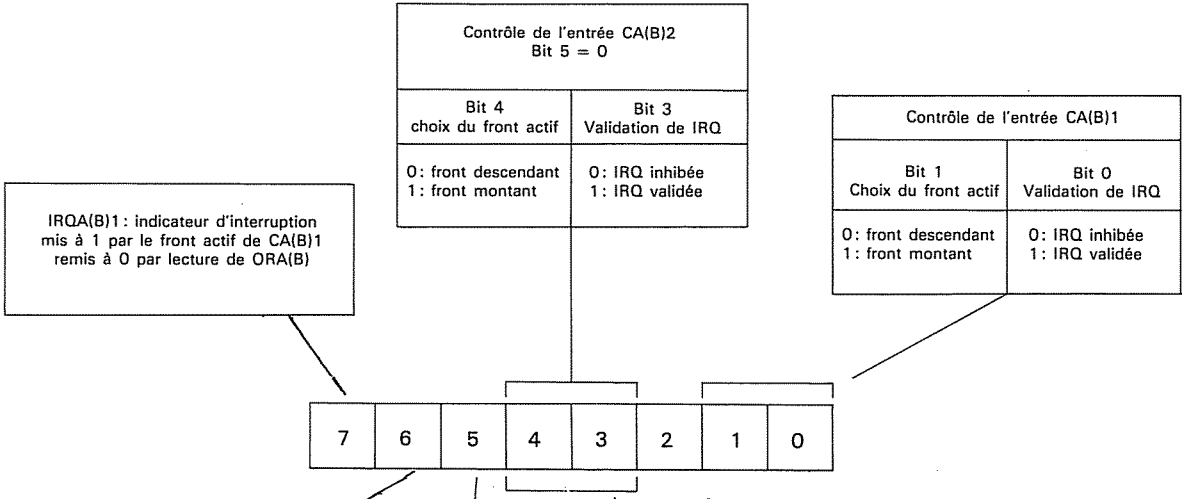
— ses registres internes sont mis à zéro,

— ses lignes d'interruption sont inhibées.

Si à présent on désire utiliser le PIA, il faut au préalable que le CPU l'initialise, c'est-à-dire qu'il lui spécifie par programme dans quelle configuration il voudrait l'avoir. Cette opération s'effectue en deux temps et un port après l'autre :

— tout d'abord on définit les directions des lignes,

— ensuite on définit le mode de fonctionnement.



Commande de CA(B)2 en sortie (Bit 5 = 1)			
Bit 4	Bit 3	CA2	CB2
0	0	impulsion déclenchée par E après lecture de ORA et restaurée par front CA1 actif	impulsion déclenchée par E après écriture dans ORB et restaurée par front OBI actif
	1	impulsion déclenchée et restaurée par E après lecture de ORA	impulsion déclenchée par E après écriture dans ORB
1		CA(B)2 suit l'évolution du bit 3	

Programmation du PIA : 6821.

Comment choisit-on si une ligne est en entrée ou en sortie ?. Il s'agit ici de voir à quoi servent les registres de direction des données.

DDRA (RESP DDRB) est un registre de 8 bits ou chaque bit indique si la ligne correspondante dans le port A (RESP port B) est une entrée ou une sortie, de la manière suivante :

Si DDRA-K (RESP DDRB-K) = 0 alors la ligne ORA-K (RESP ORB-K) sera une entrée
Si DDRA-K (RESP DDRB-K) = 1 alors la ligne ORA-K (RESP ORB-K) sera une sortie

par exemple si DDRA =	01100010B	cela voudra signifier que :
les lignes	0,2,3,4,7	du port A seront des entrées
alors que les lignes	1,5,6	seront des sorties

Pour accéder aux registres de direction des données, nous rappelons qu'il y a deux conditions à remplir :

— sélectionner correctement les lignes RS0 et RS1

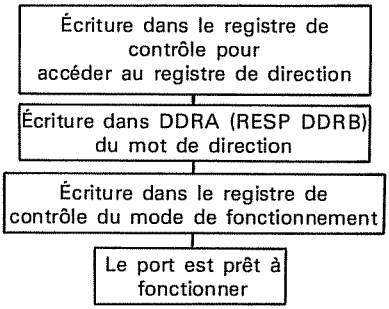
RS0 = 0 et RS1 = 0 pour DDRA
RS0 = 0 et RS1 = 1 pour DDRB

— programmer les bits CRA-2 et CRB-2 à "0".

La définition du mode de fonctionnement du PIA utilise toute la partie programmation que nous venons de voir, elle consiste à former le mot de contrôle qui définit exactement le fonctionnement désiré puis de le faire écrire par le CPU dans le registre de contrôle du port intéressé.

On donne dans la suite un organigramme qui permet de résumer la séquence d'initialisation d'un PIA.

Séquence d'initialisation d'un port de PIA



4.3. RÉALISATION PRATIQUE

4.3.1. Instructions de montage

La lourde mais nécessaire partie théorique étant maintenant terminée, nous pouvons désormais appliquer toutes ces recommandations à quelques cas particuliers. Mais au préalable il faut débuter par un petit peu de câblage. On commence par souder les broches 1 et 20 d'un support à wrapper de quarante broches sur la carte de test percée au pas de 2,54 mm, pas trop loin du connecteur. Un exemple de disposition est donné à l'annexe 4.

Ces deux broches correspondent aux alimentations du PIA et les points de soudure permettent de plaquer le support contre la carte de test. Puis si on reprend le brochage donné dans les pages plus haut on s'aperçoit qu'un certain nombre de broches ont le même nom que des broches du microprocesseur; et dans la majorité des cas lorsqu'on conçoit des systèmes électroniques à microprocesseur cela indique que ces broches doivent être connectées ensemble. C'est donc la première opération à effectuer que de wrapper une à une:

Côté PIA	Côté connecteur extensions
D0	D0
D1	D1
D2	D2
D3	D3
D4	D4
D5	D5
D6	D6
D7	D7
RESET/	RESET/

puis c'est le tour des lignes de contrôle particulières:

R/$\overline{W}$	WR/
VCC	+ 5V
VSS	0V
E	ϕ

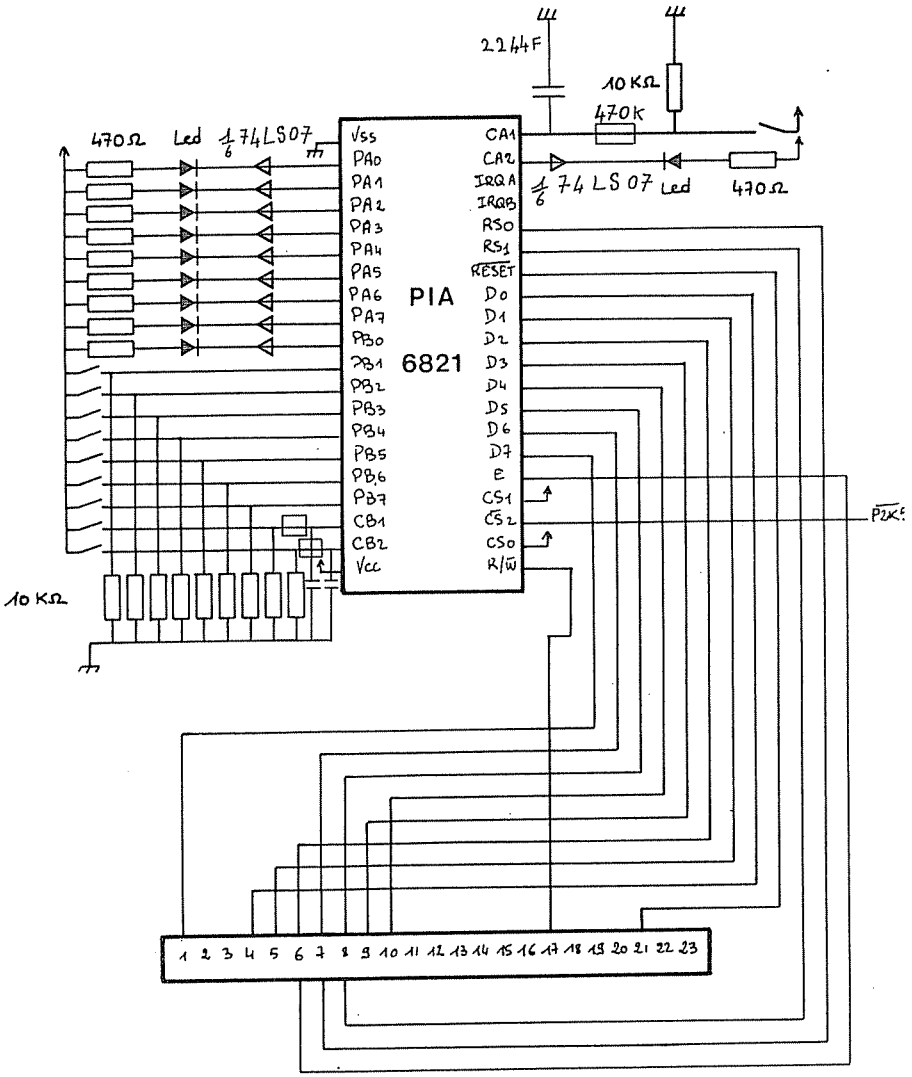
les lignes de sélection:

CS0	+ 5V	
CS1	+ 5V	
CS2/	P2K5/	← provenant du décodage d'adresses
RS0	A0	
RS1	A1	

Le boîtier PIA sera implanté à l'adresse 5 * 2 * 1024 = 10240 et donc pour les registres internes:

ORA et DDRA	seront	à l'adresse 10240
CRA	sera	à l'adresse 10241
ORB et DDRB	seront	à l'adresse 10242
CRB	sera	à l'adresse 10243

Rien ne peut remplacer un bon schéma électronique, celui-ci est donné dans les pages suivantes.

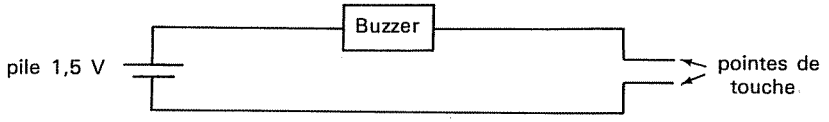


Interaçage d'un PIA (chenillard à leds).

4.3.2. Test

Quand le circuit est assemblé, il est prudent de vérifier encore une dernière fois s'il est conforme au schéma. Une méthode pratique est l'utilisation d'un indicateur de continuité qui peut être simplement un petit buzzer monter avec une pile et qui peut rendre d'incomparable service pour vérifier si les connections wrapping sont bien faites et au bon endroit.

Schéma d'un petit testeur de continuité



Vérifiez aussi précautionneusement que l'arrivée des alimentations ne sont pas inversées:

+ 5V à la broche 20
0V à la broche 1

maintenant le moment est venu d'installer le PIA 6821 sur son support; il faut forcer un petit peu pour insérer le circuit mais veiller à ne pas replier une broche par-dessous le boîtier.

Si toutes ces opérations ont été correctement suivies il ne doit pas y avoir de problème à la mise sous-tension.

Le PIA est alors dans le même circuit que le ZX 81 et on peut alors à partir du BASIC entamer un petit programme de test.

Entrez les commandes BASIC suivantes:

```
1   REM test du port d'entrée
10  LET PIA = 10240  : charge l'adresse de ORA (10240 = 2800H)
20  POKE PIA+1, 0    : sélectionne DDRA
30  POKE PIA, 0      : met le port à une entrée
40  POKE PIA+1, 4    : sélectionne ORA
50  PRINT PEEK PIA,  : lit le port A (ORA)
60  GOTO 50          : recommence à 50
```

Avant de lancer programme BASIC, attacher un conducteur métallique à la broche PA0 et garder l'autre bout dans la main.

Maintenant lancer le programme et pendant que celui-ci se déroule amener alternativement l'extrémité du conducteur métallique que vous avez dans la main au + 5V ou à la masse, ce qui revient à appliquer sur l'entrée PA0 soit + 5 volts soit zéro volt. Vous constaterez alors que parmi la série de nombres qui s'affichent sur écran on trouve:

— des nombres pairs (par exemple 254) donc leur LSB ou bit de poids le plus faible vaut zéro, tension appliquée sur PA0 par vous,

— des nombres impairs (par exemple 255) donc leur LSB vaut "1", encore une fois c'est la tension que vous avez appliquée.

Si ce n'est pas le cas, et avant d'incriminer soit l'auteur soit le composant il est conseillé de débrancher l'ensemble de système et de reprendre une à une les connexions.

Si la manipulation précédente a fonctionné, sachez que vous avez désormais un PIA en état de marche dans votre système et qu'il n'attend plus que vos ordres pour balbutier.

Nous donnons dans la suite deux petites applications rigolotes qui permettront d'abord au lecteur de se familiariser d'avantage avec ce composant et qui feront ensuite un excellent support visuel au développement de logiciel d'automatisation de tâches, discipline pour laquelle la seule limitation sera fonction de l'imagination du lecteur.

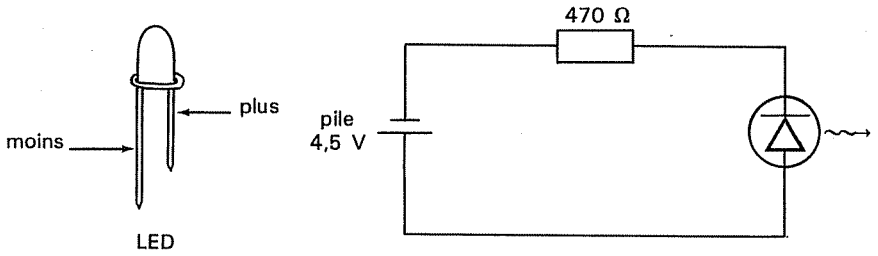
4.3.3. Application 1 : Chenillard à LEDS

Petit câblage supplémentaire

Afin de visualiser facilement les tensions présentées sur les lignes périphériques de sortie, on peut utiliser un composant peu coûteux et fonctionnant avec la même alimentation de 5 V que notre système: la LED ou diode électroluminescente.

Une LED est un composant qui émet de la lumière lorsque on la fait parcourir par un courant dans le bon sens.

Sa mise en application est simple car il suffit de l'alimenter à travers une résistance destinée à limiter le courant qui doit traverser la LED.



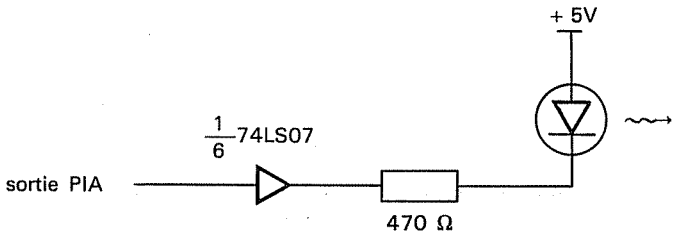
Vous pouvez monter les LED ainsi que leurs résistances de protection en les soudant sur des plateformes spéciales à cet usage et qui s'insèrent sur les supports à wrapper.

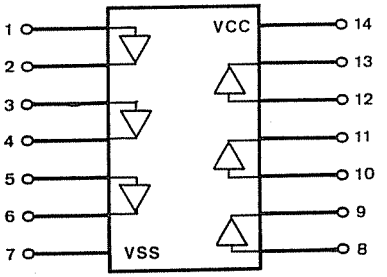
Chaque ligne du bus périphérique est alors équipé de son petit voyant: la LED est allumée quand la ligne est à "0", éteinte quand la ligne est à "1".

Il reste un petit problème: une LED nécessite pour être lumineuse au moins 10 mA et une sortie du PIA ne peut pas les fournir.

Il faut alors renforcer le bus périphérique en ajoutant un composant qui permet d'augmenter le courant de sortie. Le circuit en question est un buffer à collecteur ouvert du type 74LS07 qui comporte six portes et dont le rôle est simplement d'amplifier le courant d'entrée de chaque porte. Son brochage est donné ci-après.

L'appellation collecteur ouvert indique qu'il est obligatoire de charger les sorties utilisées, c'est-à-dire de leur mettre une résistance en série; de plus il est possible de connecter ces sorties à des sources de tension pouvant aller jusqu'à 30 V.





Hexuple buffers à collecteurs ouverts: 74LS07.

Programme

Ce petit programme restitue l'effet du chenillard de lumière, bien connu des amateurs d'ambiance disco. Le principe en est simple il suffit de disposer d'une rangée de sources lumineuses, ici les LED, qui sont à un moment donné toutes éteintes sauf une puis à l'instant suivant c'est la source contiguë qui est seule allumée et ainsi de suite... Cela donne l'impression d'une propagation de la tâche lumineuse.

Avec le PIA cela est facile à réaliser si on comprend qu'un "0" sur une ligne de données phériphériques éclairera la LED correspondante.

Alors qu'un "1" l'éteindra. C'est pourquoi il suffira d'envoyer sur le port de sortie, préalablement initialisé, un octet signifiant que la LED K doit être la seule A "0", puis de passer à la LED K+1.

C'est ce que fait très bien le programme BASIC suivant:

```
1   REM CHENILLARD A LED
10  LET PIA = 10240  : charge l'adresse de ORA
20  POKE PIA+1, 0    : sélectionne DDRA
30  POKE PIA, 255    : met le port A en sortie
40  POKE PIA+1, 4    : sélectionne ORA
50  FOR I = 0 TO 7
60  POKE PIA, (255—2**I): sortie sur port A
70  NEXT I
80  GOTO 50
```

On peut améliorer l'effet en augmentant la vitesse de défilement, à condition de programmer en langage machine; nous laisserons ce soin au lecteur.

4.3.4. Application 2 : Écho

Après s'être intéressé à un port de sortie il est logique de voir aussi les possibilités qui sont offertes avec les ports d'entrée.

La manipulation mise en œuvre au moment du test est un peu rapide, mais on peut en conserver le principe qui est d'appliquer volontairement soit 0 V soit 5 V sur les lignes d'un port d'entrée. A cet usage il est naturel de penser à des interrupteurs qui permettraient de commuter à la convenance de l'opérateur sur l'une ou l'autre des tensions d'alimentation.

C'est ce qui est réalisé dans le schéma présenté plus haut et le processus d'assemblage est le même que dans l'application 1.

L'utilisation conjuguée de LED en sortie et de switchs en entrée permet de simuler n'importe quelle conduite de processus où toutes les entrées-sorties sont sous forme logique. Ces petits ajouts au PIA peuvent devenir un outil peu onéreux pour la mise au point de logiciels débouchant sur la conduite de processus industriels.

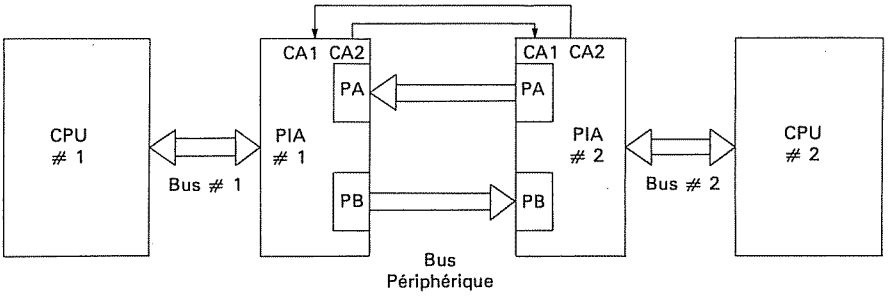
En guise d'exemple très simple, nous proposons un autre programme BASIC qui lit le port d'entrée et renvoie le contenu ainsi trouvé sur le port de sortie, ce qui permet de le visualiser avec des LED. Nous le nommons Écho.

Programme

```
1   REM ECHO              : charge l'adresse de ORA
10  LET PIA = 10240       : sélectionne DDRA
20  POKE PIA+1, 0         : met le port A en sortie
30  POKE PIA, 255         : sélectionne ORA
40  POKE PIA+1, 4         : sélectionne DDRB
50  POKE PIA+3, 0         : met le port B en entrée
60  POKE PIA+2, 0         : sélectionne ORB
70  POKE PIA+3, 4         : lit ORB et stocke dans la variable A
80  LET A = PEEK(PIA+2)   : écrit A sur ORA
90  POKE PIA,A
100 GOTO 80
```

Le petit procédé mis en évidence par cette manipulation illustre le dialogue entre deux systèmes à microprocesseur par l'intermédiaire d'un

coupleur parallèle. En effet imaginez maintenant que l'on désire coupler deux ZX 81 ensemble; il est impensable de relier leurs bus de manière directe car dans ce cas chacun des microprocesseurs demanderaient le contrôle des bus simultanément, et résultat, aucun d'eux ne pourraient fonctionner. Par contre, si dans le circuit précédent on enlève les LEDS et switchs puis on connecte, en regard du PIA d'origine, un second PIA, alors dans cette configuration on crée un bus périphérique externe aux deux systèmes qui moyennant une procédure de dialogue cohérente pourra être utilisée par l'un ou l'autre des deux micro-ordinateurs. Ce principe de liaison est assez largement utilisé dans les systèmes multi-processeurs, il est illustré dans le schéma de la page suivante.



Dialogue via PIA'S dans un système Multiprocesseur

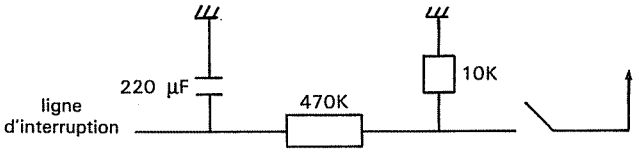
4.3.5. Application 3 : Utilisation des lignes CA1, CB1, CA2 et CB2

Afin d'imager la présentation théorique sur le fonctionnement des lignes d'interruption (CA1, CA2, CB1 et CB2), nous pensons qu'il peut être intéressant de donner quatre petits programmes BASIC permettant de les utiliser. Si nous nous reportons maintenant au schéma de câblage, nous constatons plusieurs choses:

— la broche CA1 est reliée à un interrupteur: ce sera donc une entrée,
— la broche CA2 est reliée à une LED: ce sera donc pour une sortie,
— la broche CB1 est reliée à un interrupteur: ce sera donc une entrée,
— la broche CB2 est reliée à un interrupteur: ce sera donc une entrée.

Pour CA1 et CB1 nous n'avions pas le choix, il s'agit forcément d'entrées, par contre les lignes CA2 et CB2 peuvent être programmées soit en entrée soit en sortie.

Notez aussi que les interrupteurs d'entrée sont munis de dispositifs destinés à supprimer les rebondissements conformément au schéma suivant :

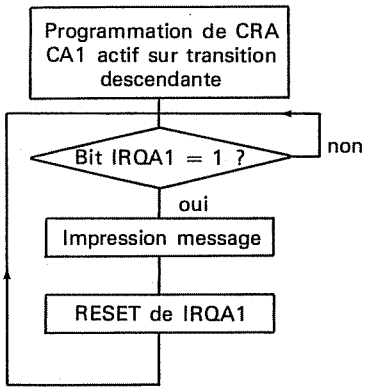


Test de CA1

On peut, toujours par programmation, choisir le front actif de cette entrée. Décidons que pour CA1, c'est la transition descendante qui sera active, ce qui signifie que lorsque la ligne CA1 passera d'un niveau haut à un niveau bas, le flag d'interruption CRA-7 se positionnera à "1" et s'y maintiendra jusqu'à ce qu'on vienne lire le registre de données ORA.

Simultanément et à condition que l'interruption soit validée, une impulsion négative sera générée sur la broche IRQA.

Afin de vérifier ce fonctionnement nous allons réaliser la séquence d'opérations définie par l'organigramme suivant :



qui est traduit par le programme BASIC :

```
1  REM TEST CA1
10 LET PIA = 10240
20 POKE PIA+1, 4          programme CRA,
                          transition descendante active de CA1
30 LET B = PEEK (PIA+1)   : lecture de CRA → B
40 IF B >= 128 THEN GOSUB 60: teste si le bit CRA-7 (IRQA 1) vaut "1"
50 GOTO 30
60 PRINT "TRANSITION DESCENDANTE DE CA1"
70 LET B = PEEK PIA       : lit ORA pour faire retomber IRQA 1
80 RETURN
```

L'essai est très simple à réaliser: actionner alternativement le switch de la ligne CA1 et de deux choses l'une :

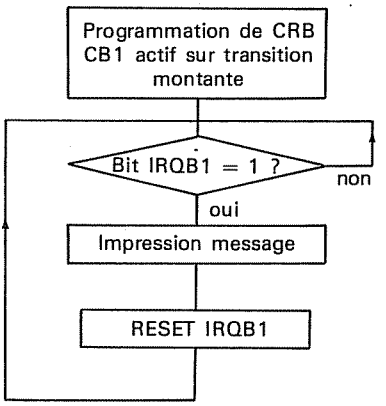
— ou bien CA1 était à "0" et passe à "1" et rien ne se produit,

— ou bien CA1 était à "1" et passe à "0" et un message apparaît sur l'écran qui indique que le PIA a reconnu un front actif de CA1.

Test de CB1

De la même manière pour la ligne CB1, mais en choisissant maintenant une transition active montante.

Le flag d'interruption CRB-7 se positionnera à "1" et s'y maintiendra jusqu'à ce qu'on vienne lire le registre de données ORB, lorsque l'état de la ligne CB1 passera d'un "0" à un "1".



qui est traduit par le programme BASIC :

```
1  REM TEST CB1
10 LET PIA = 10240+2
20 POKE PIA+1, 6          : programme CRB, transition montante
                            active de CB1
30 LET B = PEEK (PIA+1)   : lecture de CRB → B
40 IF B >= 128 THEN GOSUB 60 : teste si le bit CRB-7 (IRQB1) vaut "1"
50 GOTO 30
60 PRINT "TRANSITION MONTANTE DE CB1"
70 LET B = PEEK PIA       : lit ORB pour faire retomber IRQB1
80 RETURN
```

L'essai est réalisé comme précédemment : on actionne le switch de la ligne CB1 :

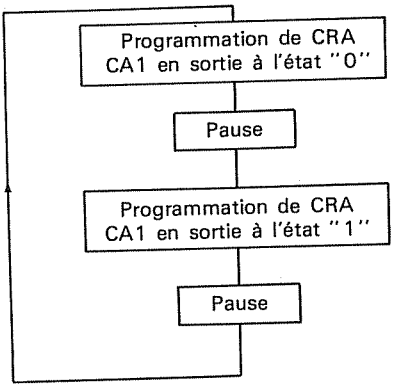
— ou bien CB1 était à "1" et passe à "0" et rien ne se produit,

— ou bien CB1 était à "0"et passe à "1" et un message apparaît sur l'écran qui indique que le PIA a reconnu un front actif de CB1.

Test de CA2

Nous avons choisi de programmer cette ligne en sortie, et plus particulièrement encore d'imposer son état par logiciel ; celui-ci pouvant varier périodiquement entre "0" et "1".

Cet enchaînement est signifié dans l'organigramme suivant :



qui est traduit par le programme BASIC très simple:

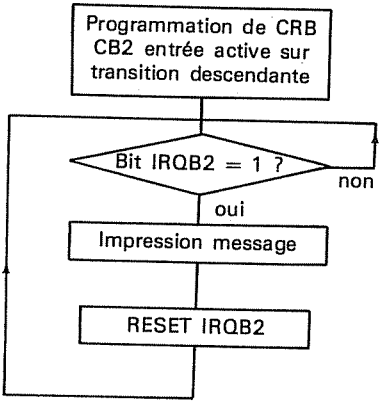
```
1   REM TEST CA2
10  LET PIA = 10240
20  POKE PIA+1, 48      : programme CRA, sortie CA2 = 0
                          (CRA-5=1, CRA-4=1, CRA-3=0)
30  PAUSE 5
40  POKE PIA+1, 56      : programme CRA, sortie CA2 = 1
                          (CRA-5=1, CRA-4=1, CRA-3=1)
50  PAUSE 5
60  GOTO 20
```

Lorsqu'on lance cette routine on constate que la LED de la ligne CA2 clignote ce qui prouve bien que l'état de CA2 évolue dans le temps comme prévu.

Test de CB2

Comme nous avions le choix et aussi pour traiter tous les cas possibles, nous décidons de programmer CB2 en entrée. Le fonctionnement est alors le même que pour les autres entrées d'interruptions CA1 et CB1.

Réalisons donc une séquence semblable:



ce qui donne en BASIC :

```
1   REM TEST CB2
10  LET PIA = 10240+2          : pointe port B
20  POKE PIA, 4                : programme CRB,
                                 CB2 entrée active descendante
30  LET B = PEEK PIA           : lecture de CRB → B
40  IF B ⩾ 64 THEN GOSUB 60    : teste si le bit CRB-6
                                 (IRQB2) vaut "1"
50  GOTO 30
60  PRINT "TRANSITION DESCENDANTE DE CB2"
70  LET B = PEEK PIA           : lit ORB pour faire retomber IRQB2
80  RETURN
```

Lancer alors le programme puis actionner alternativement le switch de la ligne CB2 et de deux choses l'une :

— ou bien CB2 était à "0" et passe à "1" et rien ne se produit,

— ou bien CB2 était à "1" et passe à "0" et un message apparaît sur l'écran qui indique que le PIA a reconnu un front actif de CB1.

Ces quatre exemples de programmation représentent en quelque sorte les actions élémentaires que peut mener votre PIA, ils seront pour vous très enrichissants sur le fonctionnement du 6821 à condition que vous décortiquiez bien, chaque fois, le contenu des registres de contrôle et leur signification.

4.3.6. Exemples d'utilisation des PIA' dans l'industrie

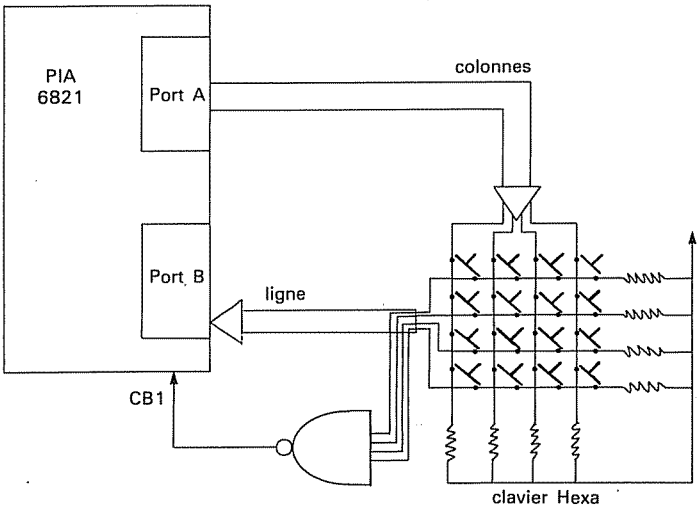
Pour aiguillonner un peu l'imagination de nos lecteurs sur les usages qu'on peut faire des PIA' , donnons assez succintement quelques principes de réalisations courantes dans l'industrie.

Décodage de clavier

La plupart du temps, pour converser avec un système informatique le moyen choisi est d'entrer les données par un clavier. Qu'il soit hexadécimal ou alphanumérique, l'organisation d'un clavier est souvent de type matricielle, cela signifie qu'il est quadrillé en plusieurs lignes et plusieurs colonnes et qu'on repère une touche comme étant à l'intersection d'une ligne et d'une colonne.

L'utilisation d'un PIA s'impose alors presque naturellement, en effet en réservant par exemple le port A la gestion des lignes et le port B à celle des colonnes, on pourra par programme en confrontant les deux informations arrivant sur le périphérique déterminer quelle touche a été enfoncée.

Le schéma de principe de la page suivante permet de mieux comprendre ce type de fonctionnement.



Décodage d'un clavier hexadécimal.

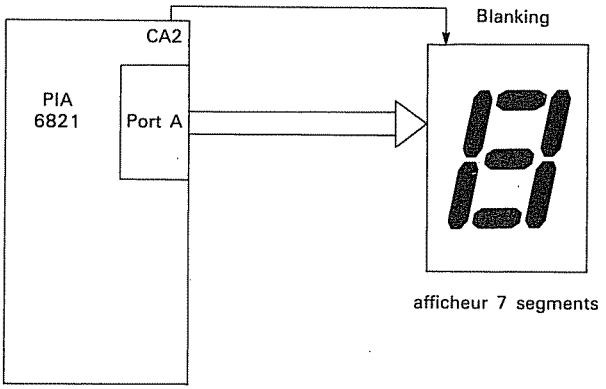
Conduite d'afficheur sept segments

L'électronique grand-public est assez friande de ce procédé de visualisation que sont les afficheurs à sept segments, il suffit de regarder la majorité des montres, réveils ou horloges qui en possèdent.

Leur principe est simple, il s'agit de sept petites sources de lumière (des LED par exemple) disposées de manière astucieuse afin de permettre de réaliser l'ensemble des chiffres hexadécimaux.

Vous entrevoyez assez immédiatement la possibilité d'utiliser un port de PIA pour piloter un afficheur de ce type; la manipulation ressemble tout à fait à ce qui a été fait plus haut avec une batterie de LEDS.

Un schéma illustrant ce principe est donné dans les pages suivantes.



Conduite d'afficheurs 7 segments

Interfaçage CAN et CNA

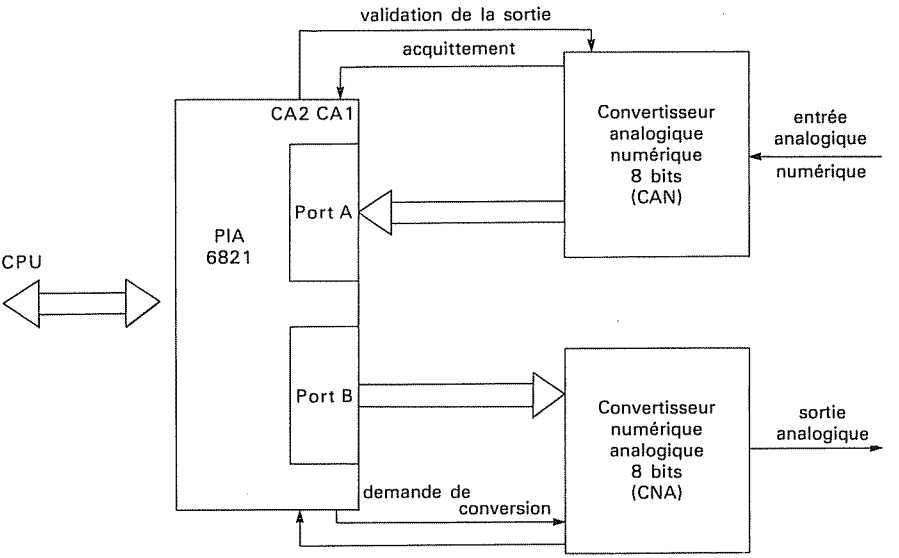
De nombreux automates industriels mettent en œuvre à un moment ou à un autre des tensions analogiques. Les valeurs de la tension électrique n'ont plus seulement une signification logique par tout ou rien: "0" ou "1".

Dans la manipulation de telles informations le microprocesseur se trouve bien impuissant. Pour remédier à cet état de fait, la solution est

d'utiliser des convertisseurs analogiques-numériques (CAN) ou numériques-analogiques (CNA) qui transforment une donnée analogique en la codant par un nombre binaire sur 8, 12 ou 16 bits que le microprocesseur sait gérer.

Afin d'interfacer ces convertisseurs avec un micro-ordinateur on intercale fréquemment entre eux un coupleur parallèle: un PIA, par exemple.

La figure ci-dessous montre un exemple de ce type d'application.



Interfaçage d'un CAN et d'un CNA.

Ce n'est pas là les seuls usages que l'on peut faire des 6821, mais notre but n'est pas d'en faire une liste exhaustive mais de familiariser le lecteur à ces techniques microprocesseurs et peut être de lui permettre d'imaginer lui-même d'autres applications. L'espace est grand ouvert, vous pouvez y aller.

4.3.7. Nomenclature des composants utilisés

Nous donnons sous la forme d'un tableau la nomenclature complète des composants utilisés dans cette extension :

Nombre	*Nomenclature*	*Description*	*Commentaire*
1	6821	PIA	
2	74LS07	Hexuple buffer à collecteurs ouverts	
10	LED		Couleur indifférente
10	Résistances 470 Ω 1/4 W		Code des couleurs : jaune, violet, marron
10	Résistances 10KΩ 1/4 W		Code des couleurs : marron, noir, orange
6	Supports 16broches	A wrapper	
1	Support 40 broches	A wrapper	
3	Tourelles 20 broches pourcomposants à souder		
3	Résistances 470KΩ 1/4 W		Code des couleurs jaune, violet, jaune
3	Condensateurs 220 NF		

5

L'extension RAM 16 koctets

5.1. PRÉSENTATION

L'utilisation d'un langage évolue comme le BASIC pose un problème de taille mémoire disponible dans le système. Le lecteur a pu se rendre compte en utilisant son système qu'il a très vite fait de saturer sa RAM 1 K lorsqu'il travaille en BASIC. Et le ZX 81 perd alors une grande partie de son intérêt. C'est pourquoi aussi toutes les extensions mémoires diffusées dans le commerce ont un tel succès. Ces extensions sont pratiquement irréprochables tant par leur consommation, que par leur encombrement ni même par leur prix. Mais un lecteur désireux d'aller plus à fond dans le fonctionnement de tels ensembles ne peut que se réjouir de trouver ici une foule de renseignements didactiques que s'il sait les exploiter pourront constituer pour lui une formation telle qu'il sera capable de concevoir lui-même des variantes à cette extension.

La grande trouvaille des fabriquants de composants qui a permis le développement d'extensions mémoires avec de telles performances est le concept de RAM dynamique.

Il faut savoir aussi qu'en électronique on assiste actuellement à une course à la mémoire et qu'un produit actuellement intéressant peut du jour au lendemain se retrouver obsolète à cause de la commercialisation d'un autre type de mémoires. Citons en vrac : les mémoires à bulles, les dispositifs à transfert de charges (CCD), les EEPROM, les NOVRAM, les RAM quasi-statiques, etc...

Dans une telle liste les RAM dynamiques font un peu office de produits dépassés, certe mais nous croyons pouvoir affirmer que c'est encore à ce jour la solution la plus économique pour réaliser une extension mémoire et c'est ce qui intéresse aussi beaucoup l'amateur.

5.2. EXPOSÉ THÉORIQUE

5.2.1. Fonctionnement

Au moment où nous avons évoqué les RAM dynamiques, nous sommes passés très vite sur les caractéristiques de fonctionnement de ces composants, il est indispensable d'y revenir avant de commencer toute manipulation les mettant en œuvre.

De manière générale les RAM dynamiques sont un peu plus difficiles à utiliser que les RAM statiques qui s'interfacent quasiment immédiatement avec le microprocesseur puisqu'elles comportent les mêmes lignes (bus adresses, bus de données et lignes de contrôle classiques).

Ces difficultés sont de plusieurs ordres :

— Les RAM dynamiques comportent quelquefois plusieurs tensions d'alimentation par exemple elles peuvent être tritension (comme le 4027 et la 4116). Et l'alimentation du système doit être prévue en conséquence.

— Ces RAM ont quasi-systématiquement une organisation par bit par exemple . 4K * 1 ou 16K * 1, mais il n'y a pas qu'une ligne de données par boîtier mais deux : une pour l'entrée des données, l'autre pour la sortie.

— Ensuite et c'est peut-être là le plus déroutant, les RAM dynamiques nécessite périodiquement un rafraîchissement de leurs données.

— Enfin les RAM dynamiques ont leur bus adresse multiplexé, cela signifie qu'une même broche réservée au bus adresse doit recevoir à deux instants différents deux lignes d'adresse différentes. L'encom-

brement d'un boîtier mémoire se retrouve réduit et la plupart des RAM dynamiques sont dans des boîtiers 16 broches.

Toutes ces caractéristiques vont faire que la conception d'une carte avec des RAM dynamiques risque d'être fondamentalement différente de ce qui est couramment fait avec les RAM statiques. Nous allons par la suite éluder une à une toutes ces petites particularités.

Nous donnons, de plus, ici un certain nombre de renseignements sur plusieurs types courants de RAM dynamiques.

Nom	*Capacité-Organisation*	*Alimentations*
4116	16 K * 1	0V, +5V, + 12V et −5V
4517	16 K * 1	0V, +5V
6632	32 K * 1	0V, +5V
6664	64 K * 1	0V, +5V
4164	64 K * 1	0V, +5V

5.2.2. L'alimentation

Dans le cas ou vous désireriez construire une extension 16K avec des 4116 vous seriez obligé de fabriquer aussi une alimentation appropriée qui viendrait prendre le relais de celle conçue pour le ZX 81.

Cette réalisation faisant l'objet d'un chapitre complet nous y renvoyons le lecteur.

5.2.3. La reconstitution du bus de données

L'organisation par bit des mémoires dynamiques oblige à en mettre huit en parallèle afin de reconstituer normalement les huit lignes du bus de données. D'autre part les entrées de données : DIN (DATA IN) peuvent être sans problème reliées au bus données du système. L'acquisition par le bloc de mémoires se faisant au moment où on applique un niveau bas sur leurs broches WR/ il semblerait naturel de relier l'ensemble de ces broches au WR/ du Z 80 ; cela pose néanmoins un problème lorsque ce signal en provenance du microprocesseur est envoyé à un autre dispo-

sitif que la RAM, par exemple le PIA de notre extension précédente, celui-ci risque d'être pris en compte par le bloc mémoire.

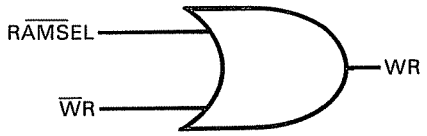
Pour mettre un terme à tous ces problèmes potentiels, il suffit de n'exciter le WR/ des mémoires dynamiques seulement lorsqu'on désire les accéder en écriture. Cela est obtenu en composant le WR/ du microprocesseur avec le signal de sélection de l'extension RAM que nous nommerons RAMSEL/, de la manière suivante:

RAMSEL/ . WR/(CPU) = WR/(RAM DYN.)

soit en se remémorant le théorème de MORGAN (cf. annexe 1):

RAMSEL + WR(CPU) = WR(RAM DYN.)

ou encore avec des portes logiques:

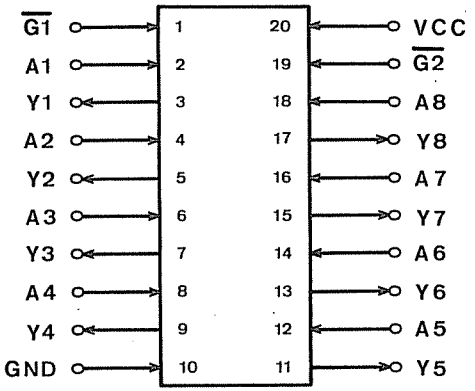


Par contre les sorties de données se stabilisent un peu après que le signal CAS/ soit apparu sur les mémoires mais il faut être sûr que ce sont bien les données valides et non pas des valeurs intermédiaires provoquées par les signaux RAS/ et CAS/.

Une solution envisageable est d'utiliser un octuple buffer trois états comme le 81LS97. Le brochage est donné ci-après. On note les boches suivantes:

— huit entrées: A1 – A8,
— huit sorties trois états: Y1 – Y8,
— deux entrées de contrôle actives au niveau bas: G1/ et G2/,
— les broches d'alimentations: VCC (+5V) et GND (0V).

Le 81LS97 va jouer le rôle d'une porte qui ne perturbe pas le bus données puisqu'il comporte des sorties trois états et qui s'ouvre quand on est sûr que la donnée amont est correctement établie.

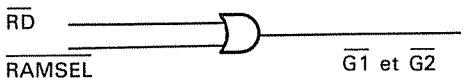


Octuple buffers trois états: 81LS97

Cette ouverture est commandée par un niveau zéro sur ses deux broches de commandes G1/ et G2/, elle doit être provoquée lorsqu'on accède au bloc mémoire dynamique en lecture. Nous entrevoyons logiquement d'appliquer à ces broches G1/ et G2/ le signal obtenu par composition des lignes RD/ et RAMSEL/ de la manière suivane:

G1/ = G2/ = RD/ . RAMSEL /
G1 = G2 = RD + RAMSEL (d'après MORGAN)

ou encore à l'aide de portes logiques:



5.2.4. Le rafraichissement

Les mémoires dynamiques stockent leurs données sous la forme d'une charge d'une petite capacité. Or une capacité (un condensateur) en électronique a toujours tendance à fuir, c'est-à-dire à perdre sa charge. Afin de maintenir une donnée correcte dans la RAM, cette charge doit être périodiquement restaurée. C'est ce procédé que l'on nomme communément le rafraîchissement.

Un cycle de rafraîchissement est accompli sur une rangée de données chaque fois qu'un cycle de lecture ou d'écriture intervient sur un bit de la rangée donnée. On parle alors de rafraîchissement par rangée seulement la durée maximale du cycle complet varie suivant le type de RAM que l'on utilise, pour fixer l'ordre de grandeur il vaut 2 ms pour la 4116.

Pour assurer que chaque rangée est rafraîchie dans le temps spécifié il faut réaliser un compteur des adresses de rangée soit de manière externe en développant un circuit logique approprié soit, comme dans certain CPU, en utilisant une fonction interne du microprocesseur. Le Z 80 est de ceux-ci, il fournit tous les signaux pour faciliter l'interfaçage avec des RAM dynamiques. En particulier le CPU effectue lui-même le comptage des rangées, qu'il mémorise dans son registre R puis qu'il envoie sur la partie basse de son bus adresse périodiquement. Et lorsque le bus adresse contient une adresse de rafraîchissement la ligne de contrôle RFSH/ passe à l'état bas.

Il serait franchement dommage de ne pas utiliser d'aussi intéressantes possibilités. Mais voyons au préalable la question suivante de multiplexage du bus adresse qui a dans sa solution une partie commune.

5.2.5. Le multiplexage du bus adresses

L'adressage d'une position mémoire de RAM dynamique doit se faire en deux étapes:

— d'abord on présente sur les broches adresses de la RAM une valeur sur sept ou huit bits qui signifie un numéro de rangée. Celui-ci est mémorisé temporairement par la RAM on dit aussi "latché" lorsque le signal spécialement réservé à cet usage: RAS/ (ROW ADDRESS STROBE) ou sélection de l'adresse de la rangée passe du niveau haut au niveau bas,

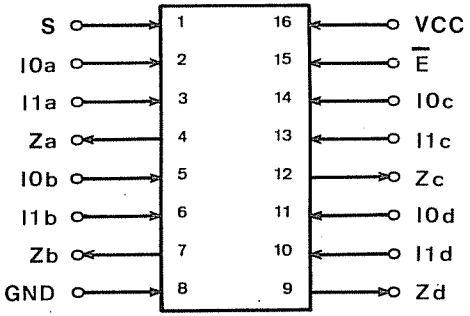
— ensuite on présente sur les mêmes sept ou huit broches adresses de la RAM une valeur qui signifie cette fois un numéro de ligne. Celui-ci est lui aussi latché sur un front descendant du signal spécial à cet usage: CAS/ (COLUMN ADDRESS STROBE) ou sélection de l'adresse de la colonne.

Les signaux CAS/ et RAS/ sont retardés l'un par rapport à l'autre, le CAS/ intervenant après le RAS/.

Nous voyons dès à présent la nécessité d'établir une petite circuiterie logique afin de permettre la génération de ces différents signaux.

L'électronique met en particulier à notre disposition un circuit intégré nommé quadruple multiplexeur 2-1 et référencé 74LS157 qui répond tout à fait à ce qu'on peut espérer. En effet celui-ci comporte:

— 8 entrées: I0A, I1A, I0B, I1B, I0C, I1C, I0D et I1D,
— 4 sorties: ZA, ZB, ZC et ZD,
— une entrée de sélection du boîtier: E/,
— une entrée de commande: S,
— deux broches d'alimentation: VCC (+ 5V) et GND (0V) (voir brochage dans les lignes suivantes)



Quadruple multiplexeurs 1-2: 74LS157

et son fonctionnement lorsqu'il est sélectionné (E/ = 0) est le suivant:

— si S = 0 alors les sorties sont identiques aux entrées avec des "0" de cette manière:

ZA = I0A; ZB = I0B; ZC = I0C et ZD = I0D

— si S = 1 alors les sorties sont identiques aux entrées avec des "1" de cette manière:

ZA = I1A; ZB = I1B; ZC = I1C et ZD = I1D

cela est résumé dans la table de vérité ci-après.

Table de vérité du 74LS157

S	E	ZK
0	0	I0K
1	0	I1K
0	1	0
1	1	0

K pouvant désigner indifféremment les parties A, B, C ou D ou démultiplexeur.

L'utilisation en parallèle de deux dispositif de ce genre sur lesquels on applique aux entrées en "0" la partie basse du bus adresse et aux entrées en "1" sa partie haute, permet de préparer le bus adresse avant d'attaquer les RAM dynamiques (voir schéma de câblage).

Il reste à voir comment nous allons générer le signal qui va commander le multiplexage et que nous nommerons MUX pour la suite.

Si nous reprenons le timing d'un cycle de recherche du code opératoire dans le Z 80 qui a été vu plus haut, nous constatons plusieurs choses :

— Celui-ci dure quatre périodes d'horloge en tout.

— Les lignes adresses contiennent pendant les deux premiers cycles l'adresse de la mémoire en liaison avec le CPU et pendant les deux derniers une adresse de rafraîchissement.

— Le signal MREQ/ retombe quand ces deux adresses sont stabilisées.

— Le signal RFSH/ est au niveau bas lorsque le bus adresse sert au rafraîchissement.

Le signal MUX doit permettre un temps de précharge des adresses basses avant de basculer sur les adresses hautes, celui-ci étant de l'ordre de 160 nano-secondes (160 10**(—9) seconde) soit environ un demi-cycle d'horloge.

En outre il doit être à l'état haut dans tous les accès au bloc mémoire que ce soit en lecture ou en écriture, cela durant la deuxième phase seulement, et à l'état bas dans toutes les autres circonstances, y compris les cycles de rafraîchissement.

Afin de réaliser une telle fonction nous allons utiliser un composant non encore cité précédemment: une bascule D, ou plus particulièrement un 74LS74 dont le brochage et la table de vérité sont donnés dans la suite.

Celui-ci comprend, pour chacune des deux bascules D qu'il contient:

— une entrée de données: D,
— deux sorties complémentées: Q et Q/,
— une entrée d'horloge: H,
— une entrée de PRESET: PR/,
— une entrée de CLEAR: CL/,
— les deux broches d'alimentation: VCC (+ 5 volts), GND (0 volt) communes aux deux bascules.

Le fonctionnement d'une bascule de ce type est conformément à la table de vérité:

— si CL/ = 0 alors quelque soit D, Q = 0 et donc Q/ = 1

ce que l'on nomme une mise à zéro:

— si PR/ = 0 alors quelque soit D, Q = 1 et donc Q/ = 0

ce que l'on nomme une mise à un:

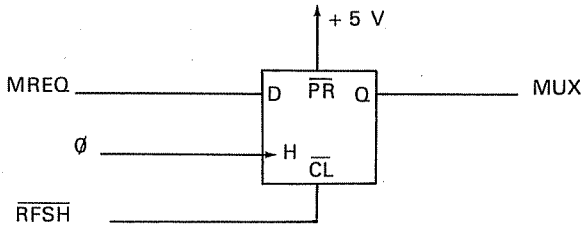
— si CL/ = 1 et PR/ = 1 alors Q reproduit fidèlement D dès la première transition montante du signal appliqué sur l'entrée H.

Pour en revenir à notre problème, et si nous câblons une bascule D de la manière suivante:

— MREQ sur entrée D,
— le signal d'horloge Ø sur l'entrée H,
— le signal RFSH/ sur le CLEAR (CL/),
— + 5 V sur PR/ pour inhiber cette entrée.

Nous obtenons ainsi sur la sortie directe Q un signal qui répond entièrement aux caractéristiques désirées pour MUX.

En effet il vaut "1" un demi-cycle d'horloge après que MREQ/ soit retombé à "0" pour un accès mémoire et est forcé à "0" par l'arrivée du signal RFSH/.



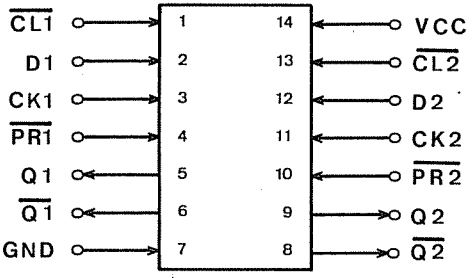
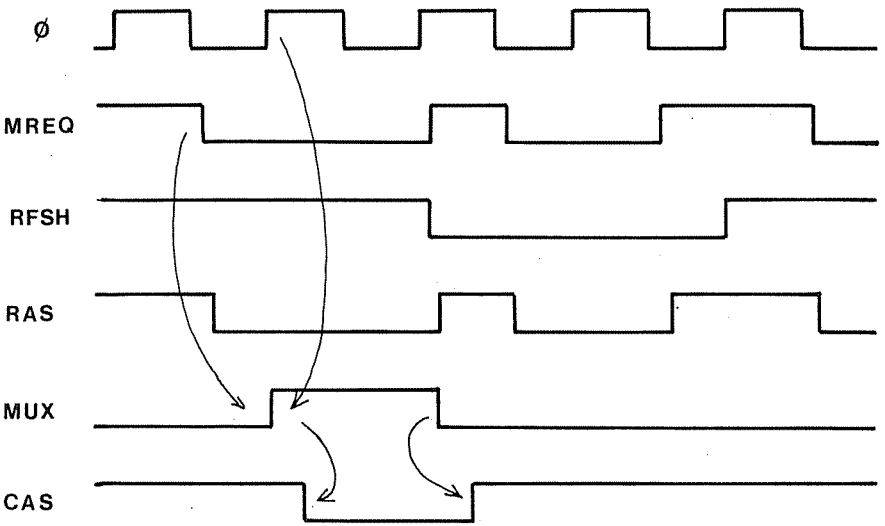


Table de vérité

PR	CL	H	Q	Q
0	X	X	1	0
X	0	X	0	1
1	1		D	D

Double bascule D : 74LS74



Timing pour RAMS dynamiques

Le diagramme de timing donné dans les pages précédentes illustre bien ce fonctionnement.

Le signal CAS/ va utiliser le MUX que nous venons de construire. Lorsque MUX passe à " 1 ", il autorise le transfert à travers les 74LS157 de la partie haute du bus adresse qui constitue alors un numéro de colonne comme nous l'avons déjà vu. Il semble approprié de profiter de cette fonction pour générer le signal de contrôle correspondant (CAS/). Seulement, toujours pour laisser aux lignes le temps de se stabiliser, nous devons retarder légèrement le signal de contrôle par rapport à l'instant d'établissement de ces adresses. Cela est réalisé par deux portes inverseuses en série sur la ligne CAS/ et qui produisent un retard d'environ 10 ns, ce qui est suffisant.

Le second signal de contrôle de sélection des adresses : RAS/ provient d'une composition un peu différente.

Nous avons vu deux cas de figure où RAS/ doit tomber à zéro :

— d'abord lorsqu'on accède au bloc mémoire, en lecture comme en écriture,

— ensuite pendant un cycle de rafraîchissement.

Suivant cette remarque on pourrait par exemple réaliser :

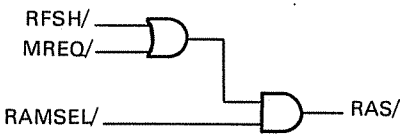
RAS/ = RAMSEL/ + RFSH/

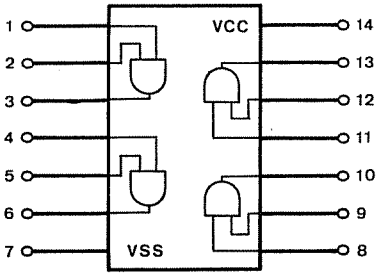
cela ne fonctionne pas puisque le signal RFSH/ intervient dès que le bus adresse va être utilisé pour transporter une adresse de rafraîchissement, avant même que les lignes soient parfaitement stabilisées (voir timing).

Pour remédier à cet inconvénient on combinera de ce cas RFSH/ et MREQ/, ainsi la relation donnant le RAS/ devient alors :

RAS/ = RAMSEL/ + (RFSH/ . MREQ/)
RAS/ = RAMSEL/ + (RSFH + MREQ)/
RAS = RAMSEL . (RFSH + MREQ) en appliquant MORGAN à deux reprises

soit encore avec les portes logiques :





Quadruple portes : AND : 74LS08

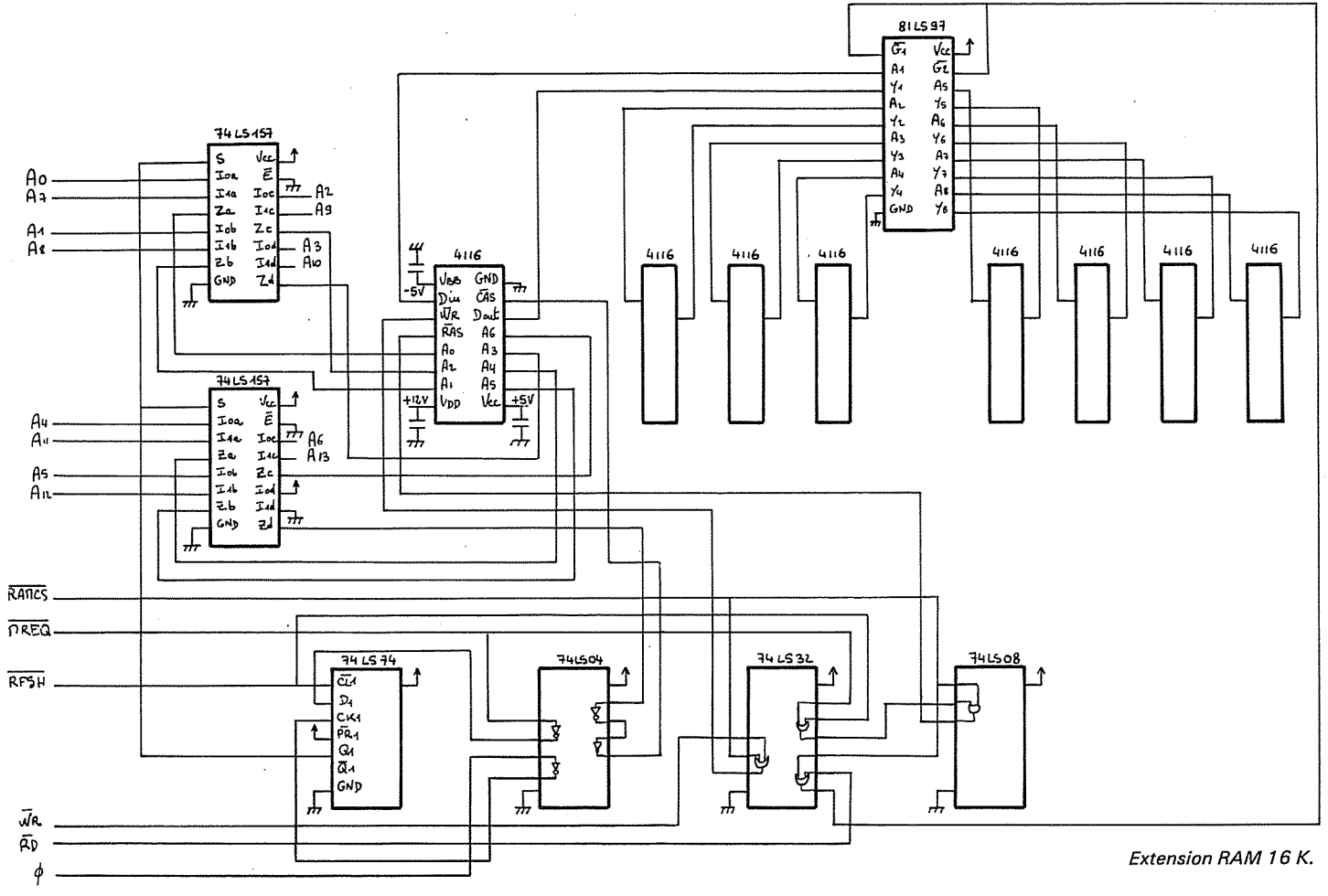
5.3. RÉALISATION PRATIQUE

5.3.1. Le câblage d'une extension RAM 16K avec des 4116-2

Toutes ces indications peuvent être utilisées pour concevoir les extensions mémoires à partir de RAM dynamiques.

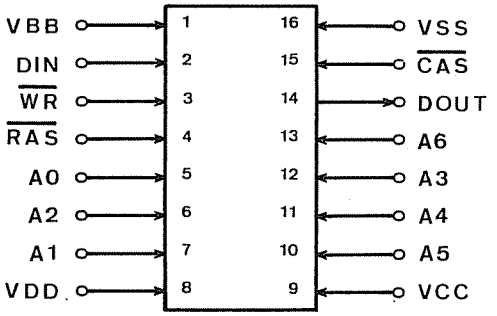
Mais pour fixer les idées nous allons proposer dans la suite l'exemple d'une extension 16K construites avec des 4116-2.

Le suffixe "–2" indique que le temps d'accès de ces dispositifs est de 200 nanosecondes. Le temps d'accès est ici, le temps que met la donnée pour se présenter sur le bus une fois que le cycle d'accès (RAS



Extension RAM 16 K.

puis CAS) est accompli. Le brochage de la 4116 est donné dans les lignes suivantes, il est classique pour une RAM dynamique.



RAM dynamique 4116.

5.3.2. Indications de montage — Wrapping

La première opération à faire est d'installer tous les supports devant accueillir les circuits intégrés. Il est préférable d'avoir les huit boîtiers RAM sur la même ligne, comme par exemple sur la disposition proposée dans l'annexe 4.

Ensuite et après avoir fixer ces supports par deux petits points de soudure comme d'habitude, on wrappe ensemble toutes les broches des supports de RAM qui ont le même numéro, à l'exception des broches de données : DIN et DOUT. C'est déjà là un travail de patience mais celui-ci doit être soigné.

Puis on relie les deux 74LS157 (quadruple multiplexeurs 1-2) au boîtier RAM le plus proche. Même chose avec le 81LS97 en faisant attention de ne pas inverser les entrées et les sorties de données.

C'est le tour des différentes portes logiques 74LS04 (six inverseurs), 74LS32 (quatre portes ou) et 74LS08 (quatre portes et); enfin on fait arriver les alimentations en prenant soin de bien les découpler, en intercalant un condensateur de 1 microfarad au tantale entre chaque tension et la masse et cela pour chaque boîtier RAM.

Toutes ces explications sont imagées dans le schéma de l'extension 16K situé dans les pages suivantes.

5.3.3. Test

Avant de mettre l'ensemble ainsi réalisé sous-tension, vérifier une dernière fois encore que ce sont les tensions correctes qui sont appliquées au bon endroit.

Dans un premier temps nous allons disposer notre nouveau bloc de RAM aux adresses allant de 32768 (8000H) à 49152 (C000H). Ce qui est facilement réalisé en utilisant comme signal de sélection (RAMSEL/) le signal précédemment établi (P16K2/) lors du décodage d'adresse. De cette manière la RAM de la version de base va pouvoir coexister avec la RAM de l'extension sans risque de conflit d'adresse.

A la mise sous-tension l'image apparaît toujours identique à ce qu'elle est habituellement. L'opérateur a entièrement le contrôle de son système et peut utiliser le BASIC sans problème.

Alors pourquoi s'en priver et demandons à BASIC de faire lui-même le test de sa nouvelle extension mémoire. L'extension, ou elle est située dans la carte mémoire n'est accessible au BASIC que par des POKES et des PEEKS, en d'autres termes on ne pourra pas implanter avec cet adressage un programme BASIC dans la nouvelle RAM.

Nous allons quand même vérifier que nous pouvons écrire et lire dans la RAM; entrez le programme BASIC suivant:

```
10 LET A = 32768                    : Adresse du début de la RAM dynamique
20 FOR I = 0 TO 255
30 POKE A+I, I                      : Écriture en RAM
40 NEXT I
50 FOR I = 0 TO 255
60 PRINT PEEK(A+I); " ";            : Lecture de la RAM
70 NEXT I
```

Lorsqu'on demande l'exécution et que tout ce passe bien, on obtient après quelques secondes, temps d'effectuer l'écriture dans les 256 premières positions mémoires de notre nouvelle RAM, la liste des nombres entiers de 0 à 255. Ce résultat est probant: il signifie que la lecture s'est bien déroulée et que votre RAM 16K est opérationnelle.

Tout autre résultat indique soit une erreur de câblage soit une défaillance d'un ou plusieurs boîtiers de mémoire dynamique.

La défaillance d'un boîtier mémoire peut à la rigueur se mettre en évidence avec un programme similaire :

```
10 LET A = 32768              : Adresse du début de la RAM dynamique
20 FOR I = 0 TO 255
30 POKE A+I, 0                : Écriture en RAM du nombre zéro
40 NEXT I
50 FOR I = 0 TO 255
60 PRINT PEEK(A+I) ; " " ;    : Lecture de la RAM
70 NEXT I
```

où on essaie de mettre à zéro les 256 premières positions de la nouvelle RAM si on obtient une autre valeur que zéro, toujours la même pendant 256 fois, c'est que les boîtiers mémoires qui génèrent sur le bus de données des bits "1" à la place de bits "0" sont abimés ; et il est facile d'identifier ceux-ci simplement en décomposant la valeur constamment obtenue en binaire.

Maintenant et dans le cas où l'extension est opérationnelle, voyons comment elle peut être installée à la place de la RAM interne de 1K.

Tout d'abord encore une petite opération du ZX 81 à cœur ouvert qui consiste à enlever les RAM internes 2114 ou 4118 de leur support, puis à dessouder la résistance R 2 de 680 Ω installée en série sur la ligne RAMCS/, afin de la remplacer par un strap ou petit bout de connecteur gainé. Ainsi la ligne RAMCS/ peut être utilisée à partir du connecteur d'extension, et on remplacera dans le rôle de RAMSEL/, le P16K2/ par ce RAMCS/.

On remonte le tout et on met sous-tension, au merveille, notre système est 16 fois plus puissant et l'image du curseur met un peu plus longtemps à apparaître car le CPU doit avant de passer le contrôle au BASIC tester non plus 1024 octets de mémoire mais 16384.

Une très bonne façon de vérifier cette affirmation est d'effectuer la commande sans doute familière au lecteur.

```
PRINT PEEK 16388 + 256 * PEEK 16389
```

et qui a pour but de connaître la taille de la RAM reconnue par le système. En fait elle exprime sur l'écran la variable RAMTOP que votre ZX 81 a mis à jour juste après avoir été mis sous-tension.

La réponse à cette question posée au microprocesseur devient avec l'extension 16 K : 32768 au lieu du 17404 obtenu sans extension mémoire.

5.3.4. Nomenclature des composants utilisés

Nous donnons sous la forme d'un tableau la nomenclature complète des composants utilisés dans cette extension :

Nombre	*Nomenclature*	*Description*	*Commentaire*
8	4116-2	RAM dynamiques	Temps d'accès 200 ns
1	74LS04	Hexuple inverseurs	Déjà utilisé dans décodage adresses
1	74LS32	Quadruple porte OR	Déjà utilisé dans décodage adresses
1	74LS08	Quadruple porte AND	
2	74LS157	Quadruple multiplexeur 2-1	
1	81LS97	Octuple buffer trois états	
24	Capacité 1 µF, 25V		Au tantale
10	Support 16 broches	A wrapper	
1	Support 20 broches	A wrapper	
13	Support 14 broches	A wrapper	

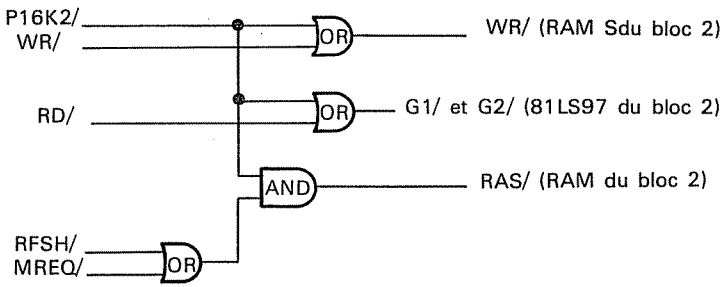
5.3.5. Extension à 32 kilo-octets

Pour les lecteurs avides de puissance nous allons donner quelques indications élémentaires qui permettront de doubler encore la capacité mémoire de votre ZX 81 et ainsi de passer allègrement à 32 kilo-octets de RAM.

Le schéma proposé pour l'extension 16K a l'énorme avantage, nous l'avons vu, d'être relogeable. Cela signifie qu'on peut l'implanter dans n'importe quelle page de 16 kilo-octets en choisissant convenablement son signal de sélection: RAMSEL/. Une conséquence directe de cette remarque est que notre extension mémoire 16K peut très bien coexister avec celle construite par Sinclair à condition bien sûr d'utiliser P16K2/ pour sélectionner la première.

Vous pouvez aussi construire d'une manière identique à ce qui a été décrit plus haut un second bloc de 16 K, toujours avec des 4116-2 et l'utiliser conjointement avec le premier. Il faut sélectionner alors l'une d'entre elle avec le signal RAMCS/ et l'autre avec P16K2/.

Il n'est cependant pas nécessaire de reproduire complètement le schéma électronique de l'extension mémoire précédente car toute la partie multiplexage du bus adresses peut être commune, et nous économisons deux boîtiers. Par contre la reconstitution du bus des données ainsi que la génération de signal RAS/ doivent être réalisées à part pour ce deuxième bloc mémoire. Cela donne aussi avec les portes logiques:



existe déjà.

Le bilan en composant de cette extension est des plus réduits. Il faut seulement rajouter :

8	4116-2	RAMS dynamiques
8	Support 16 broches	A wrapper
1	81LS97	Un octuple buffer trois états
1	Support 20 broches	A wrapper
1	74LS32	4 portes OR
1	Support 14 broches	A wrapper

L'alimentation développée dans le chapitre suivant reste toujours valable, elle a été un peu surdimensionnée à cet effet.

Disons pour finir que les 16 Koctets supplémentaires ne seront utilisables par le BASIC Sinclair qu'à condition de faire avant de commencer toute autre action :

```
POKE 16388,0
POKE 16389, 196
NEW
```

ce qui a pour effet de repousser la variable système RAMTOP jusqu'à 48 koctets, cette petite mise à jour n'étant pas faite automatiquement par le système.

Ce chapitre se termine : il contient un grand nombre d'informations qui sont assez facilement exploitables dans le but d'interfacer n'importe quelles RAM dynamiques de type courant. C'est ainsi qu'un lecteur ayant compris ces grands principes pourrait très bien envisager de construire de son propre chef d'autres extensions mémoires plus importantes encore : 48, 64 ou même encore 128 K.

6

Alimentation tritension pour ZX 81

6.1. PRÉSENTATION

Dans le cas où vous projeteriez de construire une extension de 16 koctets de mémoires avec des RAM dynamiques 4116, la première difficulté que vous devrez surmonter est de construire une petite alimentation spécialement conçue pour les 4116.

En effet, ces RAM dynamiques nous le rappelons, sont de conception assez ancienne et nécessitent pour être utilisées d'être alimenter avec trois tensions différentes : le + 5 V courant, le + 12 V et le — 5 V ce n'est pas pratique, nous le consentons mais il faut l'accepter et trouver une solution relativement économique afin de ne pas faire augmenter considérablement le prix de revient de notre extension 16 K.

6.2. APPROCHE THÉORIQUE

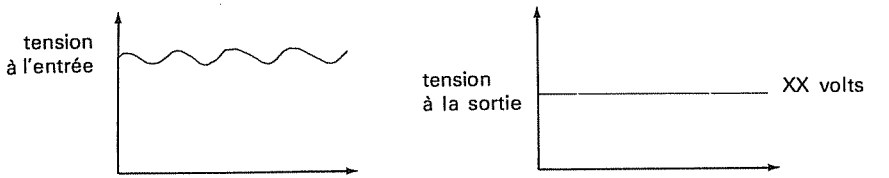
Les composants de la micro-informatique nécessitent des alimentations régulées, c'est-à-dire débarrassées au plus possible d'oscillations résiduelles pouvant provenir de l'utilisation du secteur E.D.F. Comme source primaire.

Reportez vous au chapitre sur l'alimentation du ZX 81 et vous y trouverez tous les ingrédients qui vont composer notre recette d'alimentation. En effet, le schéma de principe utilisé est général et peut, d'une

manière similaire créer n'importe quelle basse-tension positive ou négative, simplement en changeant le régulateur intégré: LM 7805.

Disons peut-être quelques mots sur ces composants: ils appartiennent à la série 78XX en ce qui concerne les régulateurs de tension positive et 79XX pour les régulateurs de tension négative. Ils sont dans un boîtier trois pattes: une pour l'entrée, une pour la sortie et la dernière pour la référence à la masse. Leur fonction est de réguler une tension redressée appliquée à l'entrée, et de fournir en sortie une tension quasiment continue à la valeur XX.

Le schéma explicite ce fonctionnement:



Sachez aussi que pour obtenir une bonne régulation il faut que la tension d'entrée soit de 3 V au moins, supérieure à la tension régulée désirée.

Leur brochage est différent suivant qu'il s'agit d'un 78XX ou d'un 79XX.

6.3. RÉALISATION PRATIQUE

Nous pensons qu'il est tout recommandé de profiter du fait que nous sommes dans l'obligation de construire une alimentation extérieure pour sortir du même coup le régulateur 7805 du boîtier du ZX 81. En effet, bien qu'étant sur un dissipateur de chaleur, celui-ci chauffe considérablement et manque un peu d'aération pour évacuer sa chaleur.

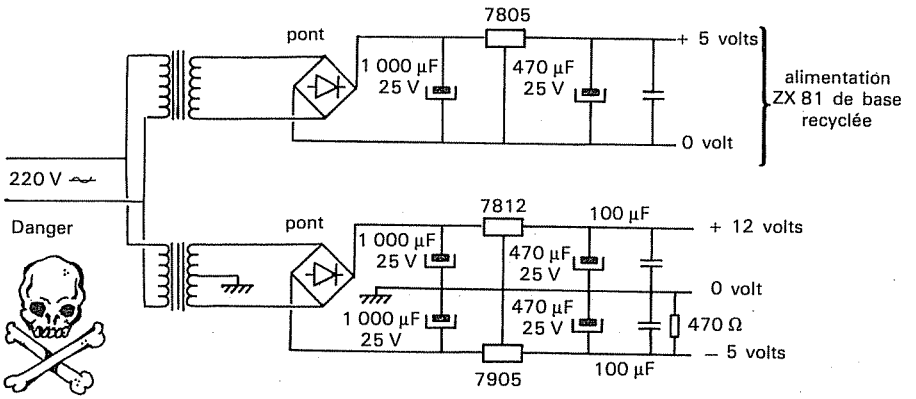
On récupère donc l'ensemble de l'alimentation ZX 81 (transformateur + régulateur + diodes de redressement + capacité de filtrage) pour l'implanter dans un autre circuit. Il n'est pas nécessaire de dessouder le transformateur, il faut simplement l'extraire, ainsi que son circuit imprimé, de son boîtier plastique. Le régulateur, lui doit être dessouder soigneusement de la carte du ZX 81, on prélèvera du même coup le radiateur formé par une plaque métallique brillante.

6.3.1. Instructions de câblage

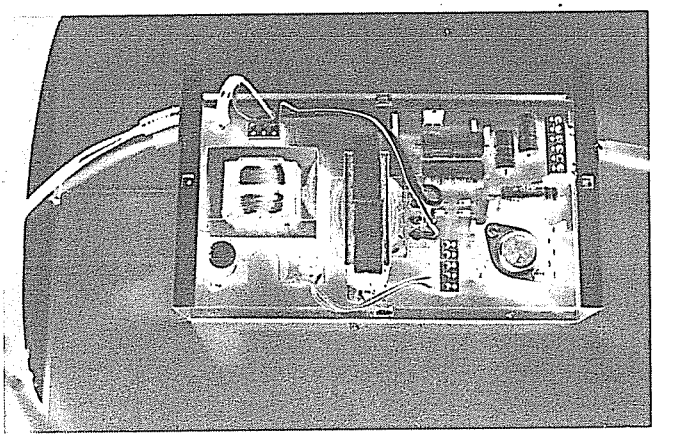
Les remarques précédentes nous ont permis d'établir le schéma électrique présenté ci-après.

On y reconnaît trois parties strictement semblables à l'alimentation ZX 81, seules les valeurs de quelques composants sont modifiées.

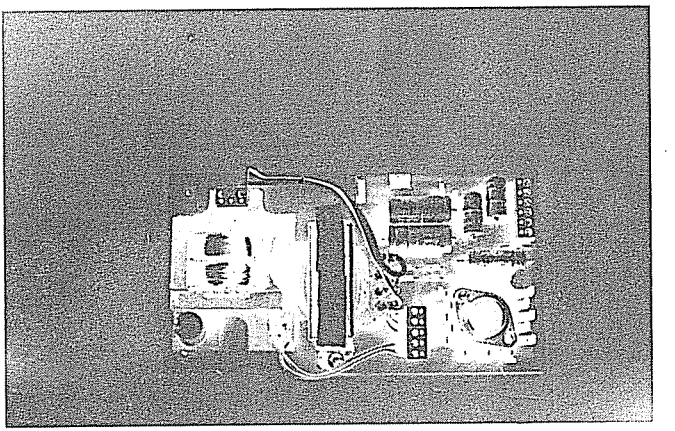
La réalisation va être menée sur un circuit imprimé, nous donnons un exemple de masque ainsi que l'implantation des composants dans les pages suivantes. L'ensemble peut être monté dans un boîtier de 200 * 120 * 60 mm.



Alimentation tritension.

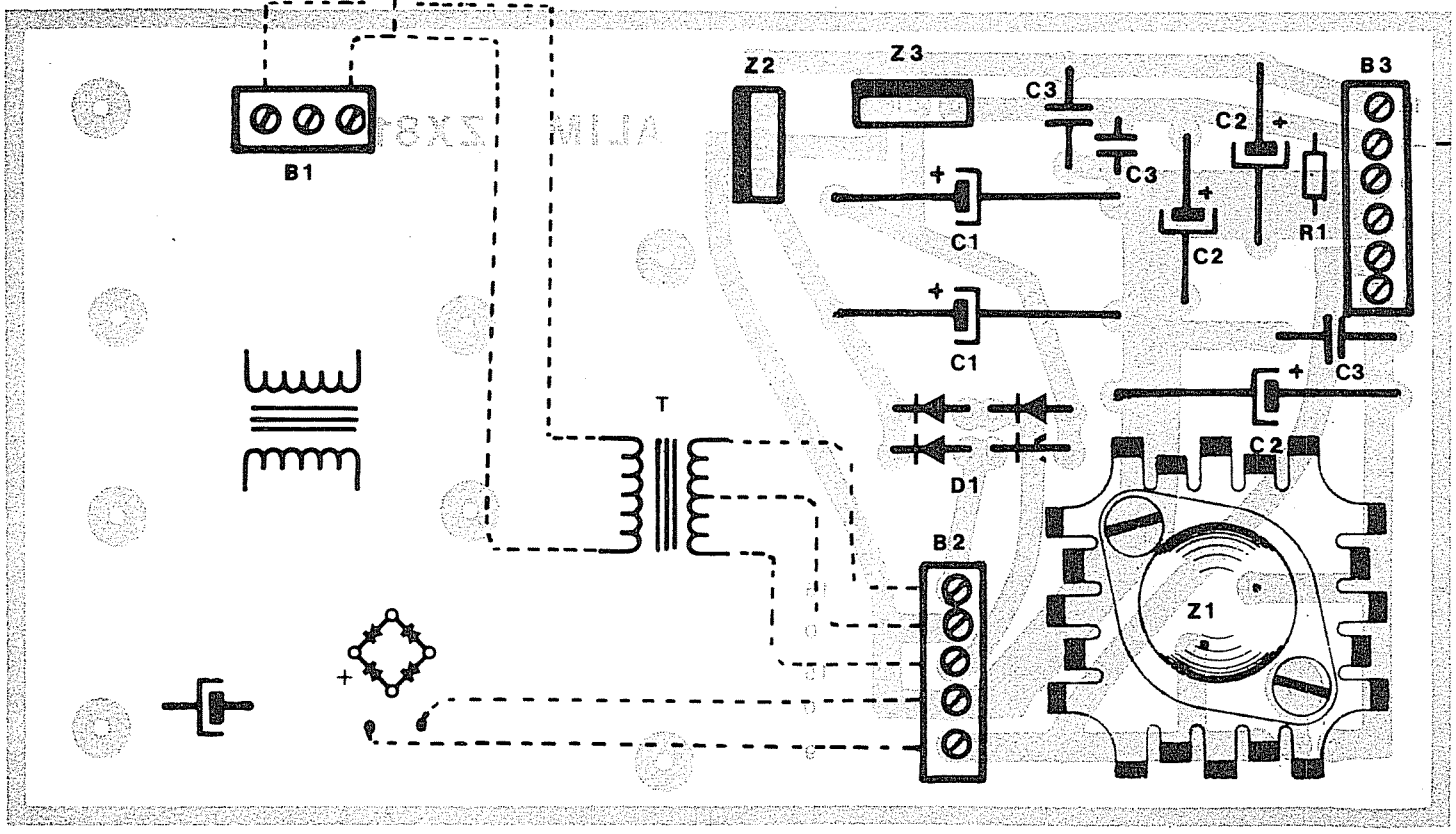


L'alimentation tritension en boîtier.



La carte alimentation tritension.

Alimentation ZX 81
Implantation des composants



Les lignes pointillées représentent les liaisons en fil souple.

6.3.2. Choix du transformateur

Compte-tenu du fait que nous recyclons le transformateur du ZX 81, il reste encore deux tensions secondaires à fournir: une pour le + 12 V et l'autre pour le − 5 V. L'idéal serait de trouver un transformateur comportant au secondaire un enroulement 15 V et un autre 8 V. Quant à sa puissance, elle est très faible puisque 16 K de RAM dynamiques 4116 consomment:

au + 12 V: 60 mA
au − 5 V: 50 µA

on voit donc qu'une puissance à partir de 3 W serait amplement suffisante et permettrait même d'utiliser ces tensions ailleurs que pour l'extension mémoire.

Nous résumons les caractéristiques souhaitées pour le transformateur:

Primaire	: 220 V
Secondaire	: Un enroulement entre 15 et 18 V Un enroulement entre 8 et 18 V
Puissance	: A partir des 3 W

6.3.3. Nomenclature des composants utilisés

Nous donnons sous la forme d'un tableau la nomenclature complète des composants utilisés dans cette extension:

Nombre	*Nomenclature*	*Description*	*Commentaire*	*Désignation*
4	1N4007	Diode		D_1
2	Capacité 1 000 µF	Chimique, 25 V		C_1
3	Capacité 470 µF	Chimique, 16 V		C_2
3	Capacité 220 µF	Mylar		C_3
1	7812	Régulateur + 12 V		Z_2
1	7905	Régulateur — 5 V		Z_3
1	2309	Régulateur + 5 V en boîtier TO-3	Pour remplacer éventuellement le 7805	Z_1
1	Radiateur pour TO-3		Pour le 2309 éventuel	
1	Transformateur	8 V et 15 V, 6 W	Voir texte	T
2	Bornes à vis	Bloc de cinq	Soudable sur carte	B_2 et B_3
1	Bornes à vis triple		Soudable sur carte	B_1
1	Résistance 470 Ω			R_1

7

Interfaçage d'un temporisateur progammable

7.1. PRÉSENTATION

Durant les nombreuses heures que le lecteur a pu passer face à face avec son ZX 81, il a dû, à plusieurs reprises, se désoler en pensant que son compagnon restait bien muet et que c'était peut être de sa part un manque certain de sociabilité.

Pour remédier à une telle situation, nous ne prétendons pas doter notre système de la capacité de parler, mais au moins de celle d'émettre des sons et même de faire un peu de musique, pourquoi pas ?

La synthèse vocale, qui est très à la mode en ce moment chez les ordinateurs, touchera un jour sans aucun doute des petits systèmes comme le ZX 81, mais pour l'instant nous nous contenterons de le faire chanter.

L'adjonction de cette nouvelle corde (vocale) à l'arc de votre système, va permettre d'accroître considérablement l'éventail de ces possibilités. Ainsi à la convenance de son programmeur le ZX 81 pourra, entre autres :

— manifester ses humeurs en émettant des bips bips tantôt agressifs tantôt encourageants,

— exécuter vos airs de musique préférés,

— synthétiser des bruits effrayants afin de sonoriser les jeux vidéo que vous avez créés ou recopiés.

7.2. EXPOSÉ THÉORIQUE

7.2.1. Le composant

La merveille de l'électronique qui va nous permettre de réaliser cette extension très facilement est un temporisateur programmable. C'est un périphérique assez couramment utilisé dans les systèmes à microprocesseur, et tout comme le PIA il est le résultat d'une intégration de fonctions très poussée, ce qui rend sa programmation un peu complexe.

La fonction du temporisateur est de générer des intervalles de temps multiples de la période d'horloge qui lui sert de référence. Ce dispositif est entièrement sous contrôle logiciel du CPU et c'est donc celui-ci qui indique la valeur initiale à partir de laquelle le temporisateur va décompter en suivant la cadence de l'horloge. Ce compte initial est programmable sur 16 bits.

De nombreux fabricants de composants commercialisent des puces de ce genre et il n'existe pas à proprement parlé de standard dans ce domaine. Cependant l'idée développée dans ce chapitre pourrait être mise en place avec n'importe quel temporisateur programmable.

Mais pour la réalisation à venir et aussi afin de fixer les idées nous avons choisi parmi toute la panoplie des produits proposés le 6840 de Motorola, encore nommé PTM pour Programmable Timer Module. Celui-ci allie des caractéristiques remarquables à un prix des plus modérés.

Il se présente sous la forme d'un boîtier de 28 broches qui intègre en fait trois temporisateurs, programmables séparément. Ces broches lui permettent de s'interfacer facilement dans la plupart des systèmes à microprocesseur; parmi elles on retrouve:

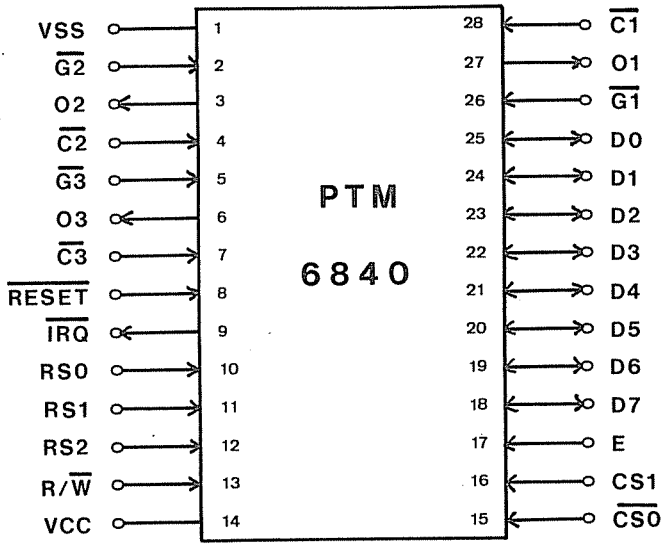
— le bus de données standard, 8 bits et bidirectionnels: D0 à D7,

— deux broches de sélection du boîtier: CS0/ et CS1,

— une entrée de lecture ou écriture: R/W/,
— une entrée de validation (ENABLE), généralement reliée à l'horloge du système: E,
— une sortie de demande d'interruption du CPU: IRQ/,
— une entrée d'initialisation externe: RESET/,
— trois lignes de sélection des registres internes: RS0, RS1_et RS2,
— trois entrées d'horloges extérieures, une par temporisateur: C1/, C2/ et C3/,
— trois sorties temporisateurs: O1, O2 et O3,
— trois entrées de validation des sorties temporisateurs, nommées aussi "GATE" ce qui signifie grille en français: G1/, G2/ et G3/,
— deux pattes d'alimentation: VCC (+ 5 V et VCC (0 V).

Le schéma donné ci-contre explicite encore d'avantage ces broches ainsi que leur direction.

Bien que notre application finale (générateur de notes) n'utilise pas les innombrables possibilités du 6840, nous pensons qu'il est nécessaire d'avoir ici un aperçu plus détaillé de l'ensemble des modes de fonctionnement de notre temporisateur, cela afin d'être en mesure de l'utiliser pleinement dans d'autres manipulations éventuelles.



7.2.2. Architecture interne

Sans trop entrer dans les détails, intéressons-nous seulement aux différents registres internes du PTM. Nous avons vu plus haut que nous disposons de trois lignes d'adresse pour sélectionner les registres internes, ce qui nous laisserait supposer qu'il y en ait au plus huit (2**3 = 8). Ceci n'est en fait pas tout à fait exact car il faut prendre en considération un fait nouveau : c'est que dans ce cas particulier du PTM certains registres sont à lecture seule et d'autres à écriture seule, soit encore lorsqu'on sélectionnera un registre pour écrire une information, la relecture au même endroit n'aura aucune raison de donner le même résultat. Pour s'y retrouver rien ne vaut le tableau suivant :

Entrées de sélection des registres			*Opération*	
RS0	RS1	RS2	R/W/ = 0	R/W/ = 1
0	0	0	Si CR2-0 = 0 écriture CR3 Si CR2-0 = 1 écriture CR1	Pas d'opération
0	0	1	Écriture registre contrôle CR2	Lecture registre d'état
0	1	0	Écriture registre tampon MSB	Lecture temporisateur #1
0	1	1	Écriture registre tampon #1 LSB	Lecture temporisateur LSB
1	0	0	Écriture registre tampon MSB	Lecture temporisateur #2
1	0	1	Écriture registre tampon #2 LSB	Lecture temporisateur LSB
1	1	0	Écriture registre tampon MSB	Lecture temporisateur #3
1	1	1	Écriture registre tampon #3 LSB	Lecture temporisateur LSB

Vous remarquerez que nous utilisons à nouveau la convention d'écriture utilisée pour le PIA : CRX-K désigné le bit K du registre de contrôle du temporisateur numéro X.

7.2.3. Programmation

Elle est assurée pour chaque temporisateur par l'intermédiaire de son registre de contrôle. La programmation a par exemple pour buts de :

— choisir l'horloge de référence ; interne ou externe,

— définir le mode de comptage du temporisateur sur 16 ou 8 bits,

— indiquer le mode de fonctionnement souhaité,
— valider les interruptions en direction du CPU,
— valider la sortie du temporisateur,
— etc...

Nous allons reprendre tout cela, en détaillant bit par bit la constitution des registres de contrôles, dont le format général est le suivant:

Bit	B7	B6	B5 B4 B3	B2	B1	B0
CR1	Validation sortie	IRQ	Mode de comptage	16/8 bits	Horloge INT/EXT	Initialisation interne
CR2	Validation sortie	IRQ	Mode de comptage	16/8 bits	Horloge INT/EXT	Accès RC1/RC3
CR3	Validation sortie	IRQ	Mode de comptage	16/8 bits	Horloge INT/EXT	Prédivision par 8

A) CRX-1

Le bit 1 de chaque registre de contrôle permet de choisirl'horloge de référence.

Si CRX-1 = 1, elle est interne, dans ce cas elle correspondra au signal appliqué sur l'entrée ENABLE (E).

Si CRX-1 = 0, elle est externe ce qui indique qu'il faudra appliquer un signal d'horloge sur l'entrée CX/ correspondante.

Cela est résumé dans la table suivante:

CRX-1	Source d'horloge du temporisateur X
0	Horloge externe entrée sur CX/
1	Horloge E

B) CRX-2

Ce bit numéro 2 indique pour chaque temporisateur si celui-ci doit travailler sur 16 ou deux fois huit bits. En effet:

Si CRX-2 = 0 alors c'est le mode normal ou le temporisateur TX compte sur 16 bits. Notons pour la suite N la valeur initiale du compteur.

Si CRX-2 = 1 le temporisateur est configuré sur deux fois huit bits. Notons dans ce cas M et L les MSB et LSB, sur huit bits, de la valeur initiale programmée dans le compteur.

CRX-2	Nombre de bits de configuration du temporisateur X
0	16 bits
1	2*8 bits

C) CRX-6

L'avant-dernier bit des registres de contrôle permet de valider ou non la sortie d'une impulsion sur la ligne IRQ/, de la manière suivante :

Si CRX-6 = 0, l'interruption est alors masquée ce qui signifie que tout critère d'interruption arrivant sur le temporisateur X n'est pas transmise au CPU par la ligne IRQ/ qui reste inactive.

Si CRX-6 = 1, au contraire, le temporisateur enverra par l'IRQ/ une demande d'interruption.

CRX-6	Validation d'interruption du temporisateur X
0	IRQ/ masquée
1	IRQ/ validée

D) CRX-7

Ces bits servent dans chaque registre de contrôle pour valider ou non les sorties OX.

Si CRX-7 = 0 alors aucun signal ne sort sur la sortie OX.

Si CRX-7 = 1 alors la sortie OX est activée.

CRX-7	Validation de la sortie compteur du temporisateur X
0	Sortie masquée
1	Sortie validée

E) CRX-0

L'indication fournit par ce bit est différente suivant le registre de contrôle dont il fait partie.

— CR1-0 indique une condition de RESET interne des trois temporisateurs :

Si CR1-0 = 0 tous les temporisateurs fonctionnent.

Si CR1-0 = 1 tous les timers sont initialisés à leur valeur initiale mais ne sont pas actifs.

— CR2-0 permet d'accéder aux deux registres de contrôle CR1 et CR3 situés à la même adresse :

Si CR2-0 = 0 l'adresse RS0 = RS1 = RS2 = 0 pointe le registre CR3.

Si CR2-0 = 1 l'adresse RS0 = RS1 = RS2 = 0 pointe le registre CR1.

— CR3-0 offre la possibilité de réaliser de manière interne une prédivision de l'horloge de référence du temporisateur #3 par huit.

Si CR3-0 = 0 le temporisateur #3 fonctionne comme les deux autres.

Si CR3-0 = 1 le temporisateur #3 est préalablement divisé par huit.

F) CRX-3, CRX-4 et CRX-5

Cet ensemble de trois bits permet de choisir le mode de fonctionnement du temporisateur. En effet le 6840 est prévu pour fonctionner dans une grande gamme d'applications. Nous pouvons résumer en considérant quatre fonctionnements principaux :

— Le mode continu où le temporisateur quand il repasse par zéro recommence inlassablement son compte à rebours à partir de la valeur programmée initialement. Il en résulte que la sortie du temporisateur émet un signal périodique dont la fréquence est égale à la fréquence de l'horloge de référence divisée par le compte initial + 1.

— Le mode Monocoup qui a un fonctionnement interne similaire au

mode précédent à part que la sortie est validée seulement pour la première impulsion.

— Le mode comparaison de fréquence ou mesure de période : si une fin de comptage apparaît avant la première transition négative de l'entrée GX/ correspondante l'indicateur individuel d'interruption du registre d'état est mis à un.

— Le mode comparaison de largeur d'impulsion fonctionne comme le mode précédent sauf que c'est une transition positive de GX/ qui termine le comptage et qui positionne l'indicateur individuel.

Le tableau suivant permet de résumer ces quatre modes ainsi que leurs flags de programmation :

Registre de contrôle			*Modes de fonctionnement*
CRX-5	CRX-4	CRX-3	
0	*	0	Continu
1	*	0	Monocoup
*	0	1	Comparaison de fréquence
*	1	1	Comparaison de largeur d'impulsion

Les bits non utilisés (*) permettent de préciser encore d'avantage les modes de fonctionnements :

Mode continu : CRX-3 = 0 et CRX-5 = 0

Le bit CRX-2, déjà nommé plus haut indique si le signal périodique issu de OX doit être symétrique ou asymétrique. Un signal périodique est dit symétrique si pendant une période donnée la durée du niveau haut est égale à celle du niveau bas, il est asymétrique sinon. On définit alors le rapport cyclique d'un signal par la quantité :

$$R = \text{rapport cyclique} = \frac{\text{largeur de l'impulsion positive}}{\text{largeur de l'impulsion négative}}$$

et dans le cas d'un signal symétrique et seulement dans ce cas, le rapport cyclique vaut "1".

De cette manière et conformément au tableau de la page suivante :

$$\text{Si CRX-2} = 0 \text{ on aura } R = \frac{N + 1}{N + 1} = 1$$

$$\text{Si CRX-2} = 1 \text{ on aura } R = \frac{M * (L + 1) + 1}{L}$$

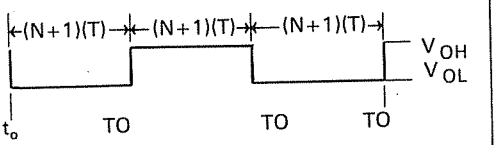
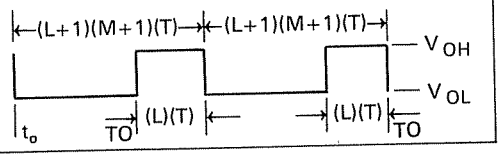
Le bit CRX-4 permet de fixer les conditions de démarrage du compteur X.

Si CRX-4 = 1 le démarrage s'effectuera soit sur un front descendant de l'entrée GX soit sur un RESET externe (impulsion négative sur la broche RESET/) ou bien interne (bit CR1-0 valant 0).

Si CRX-4 = 0 les conditions de démarrage sont les mêmes avec en plus une possibilité de démarrage immédiatement après un ordre d'écriture dans les deux registres tampons du temporisateur.

L'ensemble de la programmation du mode continu est résumé dans ce tableau :

Mode continu (CRX3=0, CRX5=0)

Registre de contrôle		*Initialisation/Formes des signaux*	
CRX2	*CRX4*	*Initialisation compteur*	*Sortie du temporisateur (OX) (CRX7=1)*
0	0	$\overline{G}\downarrow$+W+R	
0	1	$\overline{G}\downarrow$+R	
1	0	$\overline{G}\downarrow$+W+R	
1	1	$\overline{G}\downarrow$+R	

$\overline{G}\downarrow$ = Transition négative sur l'entrée G.
W = Commande d'écriture dans les registres tampons du temporisateur.
R = Initialisation du temporisateur (CR10=1 ou Reset externe=0).
N = Nombre 16 bits dans le registre tampon compteur.
L = Nombre 8 bits dans le registre tampon LSB du compteur (poids faible).
M = Nombre 8 bits dans le registre tampon MSB du compteur (poids fort).
T = Transition négative sur l'entrée d'horloge du compteur.
t_o = Cycle d'initialisation du compteur.
TO = Fin du temps de comptage.

Mode Monocoup : CRX-3 = 0 et CRX-5 = 1

La programmation détaillée est identique au mode précédent à trois exceptions près :

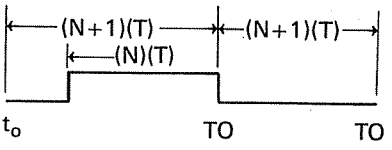
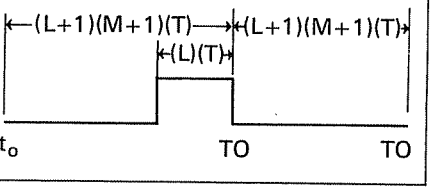
— la sortie OX est validée pour seulement une impulsion jusqu'à une réinitialisation,

— la validation du compteur est indépendante de l'entrée GX/,

— L = M = 0 ou N = 0 inhibe la sortie.

A part ces trois différences, les deux modes sont identiques :

Mode monocoup (CRX3=0, CRX7=1, CRX5=1)

Registre de contrôle		*Initialisation/Formes des signaux*	
CRX2	*CRX4*	*Initialisation compteur*	*Sortie du temporisateur (OX)*
0	0	$\overline{G}\downarrow$+W+R	
0	1	$\overline{G}\downarrow$+R	
1	0	$\overline{G}\downarrow$+W+R	
1	1	$\overline{G}\downarrow$+R	

Modes mesures d'intervalle de temps

Pour les deux derniers modes que l'on peut rassembler sous la dénomination commune de modes mesure d'intervalles de temps, le bit CRX-5 précise le sens de la comparaison ; en effet :

Si CRX-5 = 0 l'indicateur individuel d'interruption sera positionné si la durée du signal sur l'entrée GX/ est inférieure au temps de comptage.

Si CRX-5 = 1 l'indicateur individuel d'interruption sera positionné si la durée du signal sur l'entrée GX/ est supérieure au temps de comptage.

On peut résumer cela dans un tableau :

CRX3=1						
CRX4	CRX5	Application	*Condition pour mise à un de l'indicateur individuel d'interruption*	*Initialisation compteur*	*Bascule validation compteur mis à un*	*Bascule validation compteur mis à zéro*
0	0	Comparaison de fréquence	Interruption générée si la période sur l'entrée $\overline{G}$ est inférieure à la fin du temps de comptage (TO)	$\overline{G}\downarrow.\overline{I}.(\overline{CE}+TO)+R$	$\overline{G}\downarrow.\overline{W}.\overline{R}.\overline{I}$	W+R+I
0	1	Comparaison de fréquence	Interruption générée si la période sur l'entrée $\overline{G}$ est supérieure à la fin du temps de comptage (TO)	$\overline{G}\downarrow.\overline{I}+R$	$\overline{G}\downarrow.\overline{W}.\overline{R}.\overline{I}$	W+R+I
1	0	Comparaison de largeur d'impulsion	Interruption générée si la durée à l'état bas sur l'entrée $\overline{G}$ est inférieure à la fin du temps de comptage (TO)	$\overline{G}\downarrow.\overline{I}+R$	$\overline{G}\downarrow.\overline{W}.\overline{R}.\overline{I}$	W+R+I+G
1	1	Comparaison de largeur d'impulsion	Interruption générée si la durée à l'état bas sur l'entrée $\overline{G}$ est supérieure à la fin du temps de comptage (TO)	$\overline{G}\downarrow.\overline{I}+\overline{R}$	$\overline{G}\downarrow.\overline{W}.\overline{R}.\overline{I}$	W+R+I+G

Pour avoir une vision plus rapide de l'ensemble de la programmation du PTM, nous vous conseillons d'utiliser la fiche donnée plus loin.

TABLEAU 3 – PROGRAMMATION DES REGISTRES DE CONTROLE

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	1

	Registre 1	Registre 2	Registre 3
0	Tous les temporisateurs en fonction	Reg ≠ 3 peut être écrit	T3 Horloge ÷ 1
1	Tous les temporisateurs initialisés	Reg ≠ 1 peut être écrit	T3 Horloge ÷ 8

7	6	5	4	3	2	1	0
X	X	X	X	X	X	1	X

0	Horloge externe (entrée $\overline{CX}$)
1	Horloge interne (horloge E)

7	6	5	4	3	2	1	0
X	X	X	X	X	1	X	X

0	Mode compteur normal (16 bits)
1	Mode compteur deux fois 8 bits

7	6	5	4	3	2	1	0
X	X	0	0	0	X	X	X

Mode de fonctionnement continu : $\overline{G}$ ↓ ou écriture dans les registres tampons ou $\overline{Reset}$ provoque une initialisation du compteur

7	6	5	4	3	2	1	0
X	X	0	0	1	X	X	X

Mode comparaison de fréquence : interruption si $\overline{G}$ ⊓ est < fin du temps de comptage (TO)

7	6	5	4	3	2	1	0
X	X	0	1	0	X	X	X

Mode de fonctionnement continu : $\overline{G}$ ↓ ou $\overline{Reset}$ provoque une initialisation du compteur

7	6	5	4	3	2	1	0
X	X	0	1	1	X	X	X

Mode comparaison de largeur d'impulsion : interruption si $\overline{G}$ ⊔ est < fin du temps de comptage

7	6	5	4	3	2	1	0
1	X	1	0	0	X	X	X

Mode monocoup : $\overline{G}$ ↓ ou écriture dans les registres tampons ou $\overline{Reset}$ provoque une initialisation du compteur

7	6	5	4	3	2	1	0
X	X	1	0	1	X	X	X

Mode comparaison de fréquence : interruption si $\overline{G}$ ⊓ est > fin du temps de comptage

7	6	5	4	3	2	1	0
1	X	1	1	0	X	X	X

Mode monocoup : $\overline{G}$ ↓ ou $\overline{Reset}$ provoque une initialisation du compteur

7	6	5	4	3	2	1	0
X	X	1	1	1	X	X	X

Mode comparaison de largeur d'impulsion : interruption si $\overline{G}$ ⊔ est > fin du temps de comptage

7	6	5	4	3	2	1	0
X	1	X	X	X	X	X	X

0	Indicateur d'interruption masqué
1	Indicateur d'interruption validé

7	6	5	4	3	2	1	0
1	X	X	X	X	X	X	X

0	Sortie temporisateur masquée
1	Sortie temporisateur validée

NOTE : La réinitialisation peut être matérielle ou logicielle ($\overline{Reset}$ = 0 ou CR10 = 1)

Programmation du PTM 6840.

7.2.4. Le registre d'état

Comme nous l'avons vu plus haut celui-ci est accessible quand RS0 = 1 et RS1 = RS2 = 0, et cela en lecture seulement. Il contient quatre indicateurs d'interruption : trois indicateurs individuels (B0, B1 et B2) et un indicateur commun (B7). Le format du registre d'état est donc le suivant et les quatre autres bits restants (B3, B4, B5 et B6) sont inutilisés.

B7	B6	B5	B4	B3	B2	B1	B0
INT	0	0	0	0	I3	I2	I1

ou IX désigne l'indicateur d'interruption du temporisateur X et INT l'indicateur d'interruption commun.

Le bit IX sera positionné lorsque le temporisateur X aura terminé de compter ou sur les conditions d'interruptions décrites plus haut.

Le bit INT sera positionné lorsqu'un des indicateurs individuels est mis à un et que le bit 6 du registre de contrôle correspondant est aussi à un. Ce qui s'exprime par la relation logique.

INT = I1. CR1-6 + I2. CR2-6 + I3. CR3-6

7.2.5. Initialisation

Avant d'être en mesure d'utiliser le PTM il faut encore connaître la séquence d'initialisation de celui-ci.

Un niveau bas sur l'entrée RESET du PTM a plusieurs effets :

— Les registres de contrôle sont mis à zéro sauf CR1-0 (bit de réinitialisation interne) qui vaut "1".

— Le registre d'état est aussi à zéro.

— Les compteurs sont chargés à leur valeur maximale.

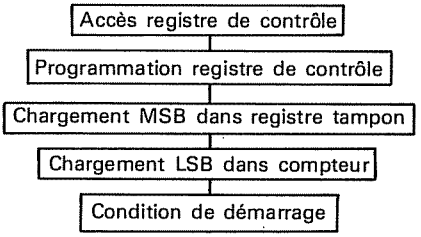
Il ne s'agit pas forcément là de la configuration désirée, et si on veut la modifier nous sommes obligés d'initialiser le PTM. Cette opération s'effectue en plusieurs étapes :

Il faut d'abord accéder au registre de contrôle du temporisateur utilisé afin de le programmer en fonction de vos besoins, il s'agit là d'une opération d'écriture banale pour le microprocesseur. Ensuite le CPU doit signifier au PTM la valeur du compte initial qu'il désire lui voir prendre en compte, et comme celui-ci est formatté sur 16 bits, cette opération se divise elle-même en deux parties :

— écrire le MSB du compte initial dans le registre tampon MSB,
— écrire le LSB dans le registre compteur du temporisateur.

Enfin le 6840 est à ce moment près à décompter il n'attend plus qu'une condition de démarrage.

On résume cette séquence d'initialisation dans l'organigramme suivant :



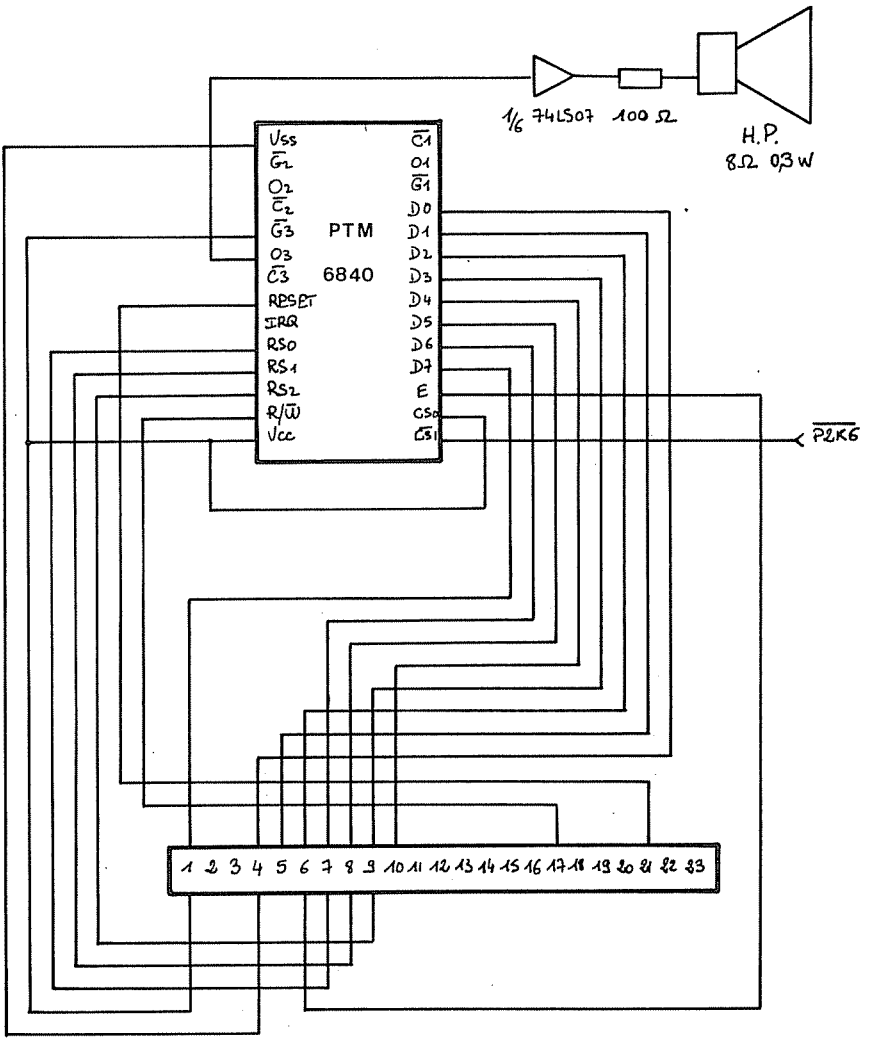
7.3. RÉALISATION PRATIQUE

7.3.1. Instructions de montage

Vous commencez à être familiarisé avec les réalisations en technique Wrapping aussi nous n'insisterons pas trop sur cette rubrique. Disons seulement que vous pouvez installer votre PTM comme il est indiqué à l'annexe 4 et que l'interfaçage de celui-ci est assez immédiat :

— Il suffit de relier le bus des données à celui du système (D0 à D7).
— Idem pour les signaux de contrôle classiques et les alimentations.

R/W/	⟶	WR/
RESET/	⟶	RESET
E	⟶	ϕ (horloge système)
VCC	⟶	+ 5V
VSS	⟶	0V



Interfaçage d'un PTM (carte sonore).

— Les broches de sélection des registres sont connectées aux adresses les plus basses de la manière suivante :

RS0	⟶	A0
RS1	⟶	A1
RS2	⟶	A2

— Et enfin, pour les broches de sélection du boîtier : on valide le CS1 constamment en le reliant au + 5 volts et on utilise le signal de sélection précédemment établi : P2K6/ pour activer le CS0/.

CS0/ ⟶ P2K6/
CS1 ⟶ + 5 volts

Toutes les broches restant en l'air sont laissées pour leur utilisation ultérieure.

7.3.2. Test — Application 1 : BIP-BIP

Le mutisme du ZX 81 n'est pas incurable, nous allons voir maintenant grâce à cette petite manipulation qui va, à la fois, servir de test du PTM.

L'organe émetteur de son est un petit haut-parleur 8 ohms, 0,3 ou 0,4 watts et avec encore un seul petit circuit intégré et une résistance supplémentaire le tour sera bientôt joué.

Le principe de fonctionnement est le suivant : grâce au PTM nous allons générer un signal périodique à une fréquence audible (entre 200 herz et 15000 herz) sur une des broches de sortie temporisateur (OX), ce signal va être amplifié légèrement pour pouvoir attaquer le haut-parleur et, c'est ce dernier qui produira un son.

Pour l'amplification du signal, comme nous ne recherchons pas des qualités HI-FI remarquables, nous nous contenterons d'utiliser un simple buffer logique en l'occurrence un 74LS07 que nous avons déjà vu, avec derrière lui en série une résistance de 100 Ω destinée à limiter le courant qui va traverser le haut-parleur. Vous pouvez vous référer au schéma donné dans les pages suivantes.

Le matériel sera tout à fait en place lorsque vous aurez relié la sortie O3 du temporisateur à l'entrée de la porte de buffer choisie pour amplifier, ainsi que l'entrée G3/ du PTM au + 5 V, de façon à la rendre inactive. Visiblement nous avons décidé de choisir le temporisateur numéro trois pour notre manipulation, cela à cause de sa propriété particulière d'être prédivisible par huit, comme nous l'avons dit plus haut.

Le temps est venu de programmer votre PTM, c'est aussi le moment d'appliquer les techniques énoncées ci-dessus.

Le PTM est sélectionné par le signal P2K6/ qui dans sa composition comporte le signal du microprocesseur MREQ/ (cf. décodage d'adresse). Cela signifie aussi en d'autres termes qu'il occupe des emplacements habituellement réservés à des mémoires, ce qui est très intéressant dans le cas présent car ces adresses sont aussi accessibles par BASIC avec des PEEKS et des POKES. Il n'y a alors aucun problème pour programmer votre 6840 à partir du BASIC.

Entrer alors le programme BASIC suivant:

```
1  REM BIP-BIP
10 LET A = 12288
20 POKE A+1, 1
30 POKE A, 0
40 POKE A+1, 0
50 POKE A, 131
60 POKE A+6, 1
70 POKE A+7, 0
80 STOP
```

Il demande un certain nombre d'explications, regardons le, ligne à ligne:

— ligne 1 : titre,

— ligne 10: on charge la variable A par la première adresse concernant le PTM, celle-ci est en relation directe avec le choix du signal de sélection P2K6/ en effet la page de 2Koctets numéro 6 commence à l'adresse:

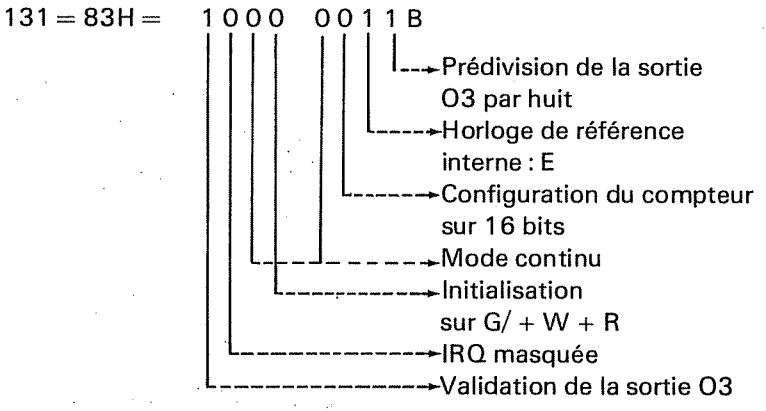
0011 0000 0000 0000B = 3000H = 12288

— ligne 20: on accède directement au registre de contrôle CR2 (RS0 = 1 et RS1 = RS2 = 0) pour positionner le bit CR2-0 à "1", ce qui a pour effet de permettre d'accéder à CR1 lorsque RS0 = RS1 = RS2 = 0.

— ligne 30: on positionne le bit 0 du registre CR1 à "0" de façon à mettre en fonction tous les temporisateurs,

— ligne 40 : on revient à CR2 pour cette fois-ci mettre "0" dans CR2-0 et cela afin d'accéder à CR3 lorsque RS0 = RS1 = RS2 = 0,

— ligne 50 : on accède maintenant au registre CR3 pour le programmer avec



— ligne 60 : on charge le MSB du compte initial dans le registre tampon, accédé lorsque RS2=RS1=1 et RS0=0,

— ligne 70 : on charge le LSB du compte initial dans le compteur ; RS2=RS1=RS0=1,

— ligne 80 : le programme BASIC est arrêté, le PTM a reçu ses ordres il fonctionne maintenant tout seul. Son démarrage s'est effectué avec l'ordre de lecture du compteur, conformément aux conditions de démarrage que nous avions programmées (condition W).

Avant de lancer ce programme, vous devriez vous soucier de savoir comment on fait pour arrêter cette délicate musique que va déverser dans vos oreilles votre ZX 81 ? En effet, nous vous défions de tenir plus de cinq minutes devant le haut-parleur, car le son émis n'est pas des plus mélodieux.

Le temporisateur une fois programmé agit tout seul sans aucun besoin du microprocesseur, et il sera inutile d'appuyer sur la touche BREAK pour arrêter le vacarme. Une méthode brutale pourrait consister à couper l'alimentation mais du même coup vous perderiez votre programme.

La bonne solution est de programmer à votre PTM de s'arrêter et vous pouvez le faire simplement en lui envoyant l'ordre BASIC:

```
POKE A, 3
```

où la variable "A" conserve la valeur qu'on lui avait donnée dans le programme: 12288. De cette manière vous intervenez sur le registre CR3 et le fait de positionner son bit 7 à zéro inhibe la sortie O3 donc il n'y a plus de signal qui va sur le haut-parleur.

Maintenant le moment est venu de lancer le programme; si un son se produit alors tout est OK, sinon il vaudrait mieux revoir votre câblage.

La hauteur du son est fonction du compte initial programmé sur seize bits dans le compteur, faites donc plusieurs essais en modifiant dans les lignes 40 et 50, les valeurs à "POKER". Pour certaines de ces valeurs il vous semblera ne rien entendre, ne paniquer pas ce n'est pas le PTM qui est en panne mais votre oreille qui est un mauvais récepteur en ce qui concerne les ultrasons.

Nous rappelons ici que le spectre des fréquences audibles se situe entre 200 et 15000 herzs, au-dessus se situent les ultrasons et en-dessous les infrasons. La relation qui lie la fréquence de sortie avec la fréquence de référence est:

$$F = FO / (8 * 2 * (N+1))$$

où FO : fréquence de référence
et N : compte initial programme dans le compteur configuré sur 16 bits

Le programme BASIC qui suit permet d'illustrer l'évolution de la fréquence de sortie sur haut-parleur lorsque N varie linéairement, le son obtenu ressemble à celui que fait un avion lorsqu'il décolle.

```
1   REM DÉCOLLAGE
10  LET A= 12288          : pointe le PTM
20  POKE A+1, 1           : écrit CR2, accède à CR1
30  POKE A, 0             : met en fonction tous les temporisateurs
40  POKE A+1, 0           : écrit CR2, accède à CR3
50  POKE A, 130           : programme CR3
60  FOR I = 0 TO 9        : boucle du MSB
```

```
70  FOR J = 0 TO 255 STEP5 : boucle du LSB
80  POKE A+6, 10—I         : charge MSB dans buffer
90  POKE A+7, 255—J        : charge LSB dans compteur
100 NEXT J
110 NEXT I
120 STOP
```

Pour les personnes désireuses de faire fonctionner aussi les deux autres temporisateurs et aussi afin de montrer par là des exemples de programmation du PTM nous donnons encore deux petits programmes en BASIC :

```
1   REM TIMER 1
10  LET A = 12288          : pointe le PTM
20  POKE A+1, 1            : écrit dans CR2, demande d'accès à CR1
30  POKE A, 130            : programme CR1
40  POKE A+2, 255          : charge MSB dans registre tampon
50  POKE A+3, 255          : charge LSB dans compteur 1
60  PRINT PEEK(A+2)        : écrit le MSB du compteur 1
70  PRINT PEEK(A+3)        : écrit le LSB du compteur 1
80  GOTO 60
```

```
1   REM TIMER 2
10  LET A = 12288          : pointe le PTM
20  POKE A+1, 1            : écrit CR2, accède à CR1
30  POKE A, 0              : programme CRA, met en fonction tous les
                             temporisateurs
40  POKE A+1, 131          : programme CR2 (accès direct)
50  POKE A+4, 255          : charge MSB dans registre tampon
60  POKE A+5, 255          : charge LSB dans compteur 2
70  PRINT PEEK(A+4)        : écrit le MSB du compteur 2
80  PRINT PEEK(A+5)        : écrit le LSB du compteur 2
90  GOTO 70
```

Ces programmes donnerons des résultats satisfaisants seulement si les entrées G1/ et G2/ sont reliées au + 5 V.

Les lignes 60 et 70 permettent de constater la progression du compteur en lisant celui-ci à des instants différents. En effet il apparaît sur l'écran une série de nombres qui semblent complètement aléatoires mais qui en fait représentent les états successifs du compteur à chaque lecture de ce dernier.

7.3.3. Application 2 : Générateur de bruit

Nous allons maintenant exploiter la possibilité de modifier le compte initial pour synthétiser des bruits de manière automatique.

La première idée est d'utiliser une séquence d'octets aléatoire pour charger le compteur. Nous disposons d'une telle séquence il s'agit de la ROM; en effet supposez que vous programmiez le PTM avec les deux premiers octets de la ROM puis avec les deux suivants et ainsi de suite jusqu'à épuisement (du lecteur) vous générez de cette manière un bruit qui n'a rien de musical mais qui permet d'illustrer le fonctionnement de notre extension de manière amusante.

C'est ce que fait automatiquement le programme BASIC suivant :

```
1  REM ROM MUSICALE
10 LET A = 12288          : pointe le PTM
20 POKE A+1, 1            : écrit CR2, accède à CR1
30 POKE A, 0              : programme CR1, active tous les
                            temporisateurs
40 POKE A+1, 0            : écrit CR2, accède à CR3
50 POKE A, 130            : programme CR3
60 FOR I = 0 TO 8192      : boucle sur les adresses de la ROM
70 POKE A+6, PEEK I       : charge MSB dans tampon
80 POKE A+7, PEEK (I+1)   : charge LSB dans compteur
90 NEXT I
```

Après la génération de sons aléatoires voici une manière beaucoup plus systématique de faire de la création musicale. Nous imitons le bruit

de la sirène en effectuant une modulation sonore alternativement ascendante puis descendante.

```
1   REM SIRÈNE
10  LET A = 12288          : pointe le PTM
20  POKE A+1, 1            : écrit CR2, accède à CR1
30  POKE A, 0              : programme CR1, active tous les
                             temporisateurs
40  POKE A+1, 0            : écrit CR2, accède à CR3
50  POKE A, 130            : programme CR3
60  LET J=0
70  LET U=1
80  LET J=J+5*U
90  POKE A+6, 6
100 POKE A+7, 100+J
110 IF(J=OOR J=50)
      THEN LET U=—U
120 GOTO 80
```

7.3.4. Application 3 : Générateur de notes

Toujours plus compliqué, vous entrevoyez peut-être déjà la possibilité de calibrer les fréquences programmées afin de constituer un générateur de notes.

Auparavant ayons quelques petites considérations physiques sur la musique. Le "LA" pur donné par un diapason est une oscillation harmonique à 440 Hz. Pour obtenir le "LA" de l'octave immédiatement supérieure il faut multiplier par deux la fréquence on obtient ainsi: 880 Hz.

Cela est général, ainsi pour passer à l'octave supérieure il suffit de multiplier la fréquence de la note par deux. C'est une loi multiplicative.

Si maintenant on considère que dans une octave il y a douze demi-tons qui sont équitablement repartis en fréquence, on déduit assez naturellement que le passage d'un demi-ton à l'autre s'obtient un multipliant

la fréquence du premier par la racine douzième de deux soit encore :

$$\sqrt[12]{2} = 1,05946$$

La gamme chromatique, c'est-à-dire demi-ton en demi-ton, est la suivante :

DO, DO#, RE, RE#, MI, FA, FA#, SOL, SOL#, LA, LA#, SI, DO

Lorsqu'on va des graves vers les aigus, ce qui correspond aussi à une augmentation de la fréquence.

Si maintenant on se fixe une origine au LA à 440 Hz on calcule les fréquences :

- pour le LA# par 440 * 1,05946 = 466,16
- pour le SOL# par 440 / 1,05946 = 415,30

et ainsi, pour toutes les notes que l'on désire on peut faire la correspondance note-fréquence.

Le temporisateur, de son côté, sort un signal, nous l'avons vu plus haut, à la fréquence de :

F = 3 250 000 / (8 * 2 * (N+1)) = 203 125 / (N+1) Herzs

et c'est en choisissant convenablement la valeur du rapport de division N que nous pourrons calibrer les notes.

Toutes ces remarques nous ont permis de dresser une table qui fait la correspondance entre le nom des notes, lèur fréquence, le rapport de division nécessaire et la décomposition en deux octets afin de préparer la programmation (N = M * 256 + L).

Table de correspondance

Note	Fréquence (Hz)	Rapport de division	MSB	LSB
LA	440	461	1	205
LA#	466,16	435	1	179
SI	493,87	410	1	154
DO	523,27	387	1	131
DO#	554,36	366	1	110
RE	587,31	345	1	89
RE#	622,22	326	1	70
MI	659,22	307	1	51
FA	698,43	290	1	34
FA#	740,01	273	1	17
SOL	783,95	258	1	2
SOL#	830,60	243	0	243
LA	880	230	0	230
LA#	932,32	217	0	217
SI	987,76	205	0	205
DO	1 046,50	193	0	193

Cette table a été dressé en utilisant les possibilités de calcul du ZX 81, à l'aide du programme BASIC :

```
1   REM CALCUL
10  LET F0=203125
20  LET F1=440
30  PRINT TAB 2;"FREQ";TAB 13;"N";TAB 19;"M";TAB 27;
    "L"
40  PRINT
50  FOR I = 0 TO 15
60  LET F=INT(F1*(1.05946**I)+0. 5)
70  LET N=INT(F0/F-0. 5)
80  LET M=INT(N/256)
90  LET L=INT(N-256*M+0. 5)
100 PRINT F; TAB 8; N; TAB 16; M; TAB 24; L
110 NEXT I
120 STOP
```

Nous pouvons dès à présent utiliser toutes ces données un petit peu techniques pour réaliser un petit générateur de musique qui utiliserait les touches numériques du ZX 81 (touches de 1 à 9)

Le programme BASIC qui suit réalise cette fonction:

```
1     REM ORGAN
10    LET A = 12288          : pointe le PTM
20    POKE A+1, 1            : programme CR1 pour activer tous
                               les temporisateurs
40    POKE A+1,0             : écrit CR2, accède à CR3
50    IF INKEY$<>"" THEN GOTO 90
60    POKE A, 3              : inhibe la sortie O3
70    LET J=0
80    GOTO 50
90    LET K=CODE INKEY$-28
100   IF K=J THEN GOTO 50    : compare avec la valeur de la
                               touche précédente
110   LET J=K                : mémorise dans J l'état de la
                               touche prise en compte
120   IF(K<1) OR (K>9) THEN GOTO 50
130   POKE A, 131            : programme CR3, active la sortie
                               O3
140   POKE A+6, M(K)         : charge MSB
150   POKE A+7, L(K)         : charge LSB
160   GOTO 50

1000  REM PROGRAMME DE CHARGEMENT
1010  PRINT "NOMBRE DE NOTES?";
1020  INPUT N
1030  PRINT N
1040  DIM M(N)
1050  DIM L(N)
1060  FOR I = 1 TO N
1060  PRINT "NOTE";I; "MSB =";
1070  INPUT M(I)
1080  PRINT M(I); "LSB =";
1090  INPUT L(I)
1100  PRINT L(I)
1110  NEXT I
9999  STOP
```

Il demande un certain nombre d'explications:

— Les lignes 10 à 40 initialisent le temporisateur 3.

— La ligne 50 permet de reconnaître si une touche a été enfoncée sur le clavier.

— Les lignes 60 à 80 arrêtent le temporisateur et boucle sur la surveillance du clavier.

— Les lignes 90 à 120 vérifient que la touche enfoncée a changé par rapport à la fois précédente, et qu'elle correspond bien à une touche numérique.

— Les lignes 130 à 160 chargent les compteurs avec son rapport de division, activent ce dernier puis retournent à la surveillance du clavier.

— Les lignes 1000 à 9999 constituent un programme classique de chargement des variables, en l'occurence des comptes initiaux à programmer.

Pour utiliser ce programme il faut exécuter le chargement en faisant:

```
RUN 1000
```

L'ordinateur pose alors des questions auxquelles il faut répondre:

```
Nombres de notes? 9
Note 1 MSB = 1      LSB = 205
Note 2 MSB = 1      LSB = 154
Note 3 MSB = 1      LSB = 131
Note 4 MSB = 1      LSB =  89
Note 5 MSB = 1      LSB =  51
Note 6 MSB = 1      LSB =  34
Note 7 MSB = 1      LSB =   2
Note 8 MSB = 0      LSB = 230
Note 9 MSB = 0      LSB = 205
9/9999
```

A ce niveau il serait sage de sauvegarder ce programme sur cassette. Enfin pour demander l'exécution veillez à taper: GOTO 1 et non pas RUN sinon toutes les variables que vous venez d'entrer seraient effacées et vous n'auriez plus qu'à recommencer le chargement.

Le clavier est maintenant sensible, et vous pouvez exécuter vos airs favoris en utilisant les touches numériques qui ont les valeurs suivantes :

Touche	Note
1	LA
2	SI
3	DO
4	RE
5	MI
6	FA
7	SOL
8	LA
9	SI

Remarquez que ce programme constitue un exemple et que vous lecteur vous avez désormais en main toute les informations pour étendre votre orgue à plusieurs octaves.

Toujours avec ce même principe nous pouvons exécuter un air préalablement mis en mémoire.

Entrer le programme BASIC suivant, l'effet est saisissant :

```
1   REM MORCEAU CHOISI
10  LET A = 12288          : pointe le PTM
20  POKE A+1, 1            : écrit CR2, accède à CR1
30  POKE A, 0              : programme CR1 pour activer les
                             temporisateurs
40  POKE A+1, 0            : écrit CR2, accède à CR3
50  FOR I=1 TO N
60  POKE A, 131            : programme CR3, active compteur
70  POKE A+6, M(I)
80  POKE A+7, L(I)
90  PAUSE 5
100 POKE A, 3              : programme CR3, inhibe sortie
110 PAUSE 5
120 NEXT
130 GOTO 50
```

et toujours en utilisant le programme de chargement précédant :

```
Nombres de notes? 11
Note 1   MSB = 1     LSB = 131     . DO
Note 2   MSB = 1     LSB = 131     . DO
Note 3   MSB = 1     LSB = 131     . DO
```

```
Note 4   MSB = 1    LSB =  89    .RE
Note 5   MSB = 1    LSB =  51    .MI
Note 6   MSB = 1    LSB =  89    .RE
Note 7   MSB = 1    LSB = 131    .DO
Note 8   MSB = 1    LSB =  51    .MI
Note 9   MSB = 1    LSB =  89    .RE
Note 10  MSB = 1    LSB =  89    .RE
Note 11  MSB = 1    LSB = 131    .DO
```

Vous lancez la procédure par GOTO 1.

En conclusion de ce chapitre, disons que l'usage qui est fait ici du PTM est relativement limité, mais cela nous a permis de nous familiariser à l'utilisation de ce composant. Nous pensons avoir donné suffisamment d'informations pour permettre à des amateurs de développer toutes les idées naissant du fin fond de leur imagination et pouvant mettre en œuvre le 6840.

7.3.5. Nomenclature des composants utilisés

Comme d'habitude nous donnons en fin de chapitre la nomenclature complète des composants utilisés dans cette extension.

Nombre	*Nomenclature*	*Description*	*Commentaire*
1	6840	Temporisateur programmable PTM	
1	74LS07	Hexuple buffer à collecteurs ouverts	Déjà utilisé dans interfaçage d'un PIA
1	Support 28 broches	A wrapper	
1	Support 14 broches	A wrapper	
1	Haut-parleur 8 Ω		Environ 0,3 W
1	Résistance 100 Ω		Code des couleurs marron, noir, marron

8

Interfaçage d'une EPROM

8.1. INTRODUCTION

Il ne fait aucun doute que le ZX 81 prend petit à petit la mesure d'un phénomène social. De plus en plus de gens désirent mettre à profit de tout le monde leurs trouvailles et il s'agit pour la plupart du temps d'amélioration de logiciel.

D'un autre côté, la programmation en assembleur devient tout à fait accessible au grand public grâce à une multitude d'ouvrages traitant de ce sujet. Ces deux remarques nous amènent à entrevoir une nouvelle possibilité d'améliorer encore les performances de notre système, en prévoyant dès à présent, des espaces disponibles de mémoire, destinés à accueillir un supplément de logiciel résident, c'est-à-dire implanté en mémoire morte et plus exactement sur des EPROM.

En effet, à notre avis, il se peut très bien que dans un avenir tout proche, des logiciels sur EPROM pour le ZX 81 soit disponibles dans le commerce. Et quel amateur ne rêve pas d'avoir un jour directement implantée sur son système une foule de routines utilitaires qui lui permettraient de disposer de fonctions plus élaborées comme par exemple :

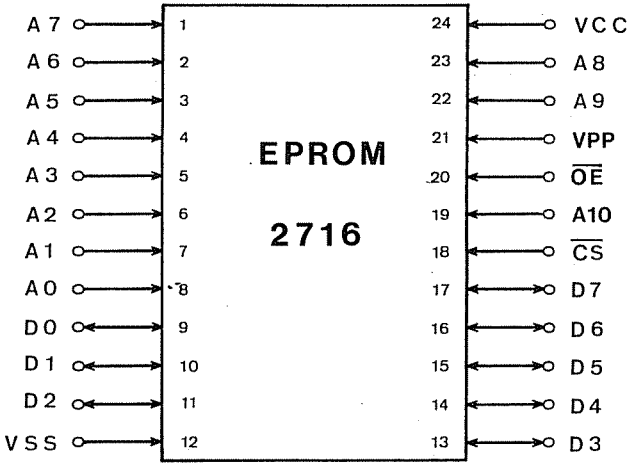
— le graphisme haute résolution,
— un moniteur d'exploitation,
— un assembleur résident,
— etc...

Afin d'être prêt à affronter cette nouvelle petite révolution, donnons dès à présent quelques directives sur l'utilisation des EPROM et en particulier de la 2716 commercialisée par Intel, Motorola et bien d'autres fabricants de composants.

8.1.1. Description et brochage du composant

L'EPROM en question se présente sous la forme d'un boîtier standard de 24 broches. La particularité des boîtiers EPROM est de comporter sur leur dessus une petite loupe à travers laquelle il est possible de voir le circuit microscopique. La 2716 a une capacité de 2 kilo-octets et une organisation, 2K * 8, cela implique tout naturellement qu'elle comporte :

— huit lignes de données D0 à D7,
— onze lignes d'adresses A0 à A10 (2 ** 11 = 2048).



Avec en plus quelques lignes de contrôle :

— OE/ : Entrée de validation des sorties,
— CS/ : Entrée de sélection du boîtier,
— VPP : Entrée réservée à la programmation.

L'alimentation est en 0 et + 5 V et elle s'applique respectivement sur les broches 12 (GND) et 24 (VCC).

Le brochage complet est donné ci-dessus.

8.1.2. Interfaçage

L'interfaçage d'une EPROM ne présente pas de difficultés particulières si ce n'est que lorsqu'elle est dans le système il est conseillé d'appliquer le + 5 V sur l'entrée VPP de la 2716.

La broche OE/ (OUTPUT ENABLE) est aussi généralement reliée au RD/ du système, ce qui a pour effet de valider les broches de données en sortie lorsqu'on accède à l'EPROM en lecture. Nous rappelons qu'une EPROM ne peut pas être accédée en écriture.

Le schéma de la page suivante explicite ces instructions.

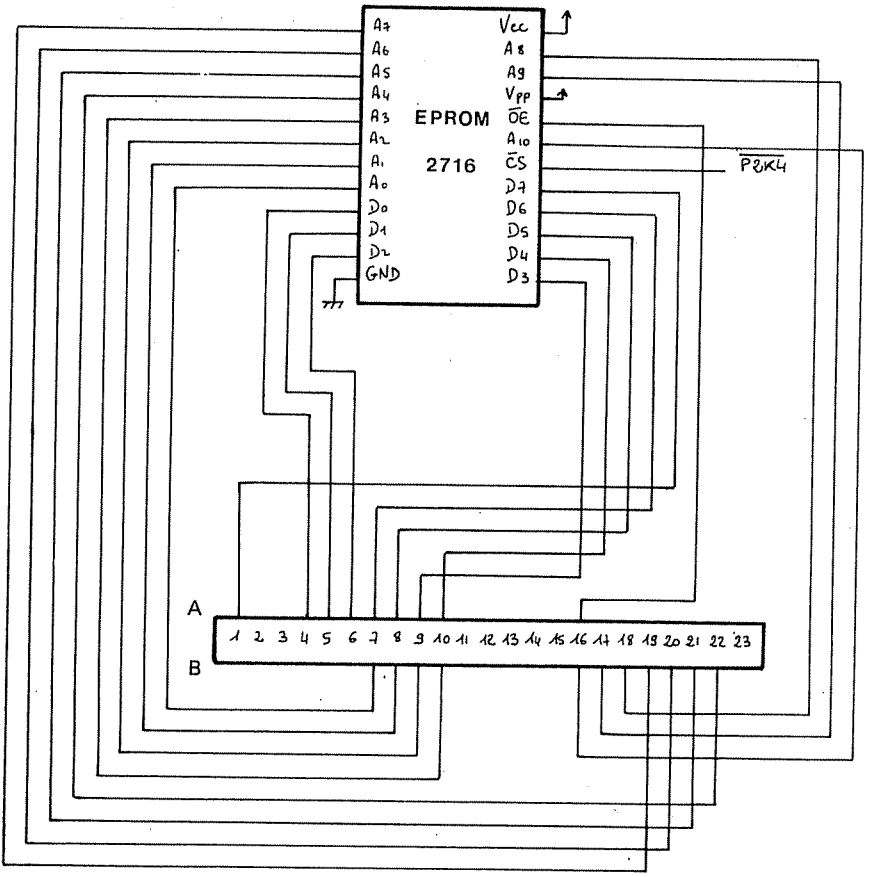
8.1.3. La programmation

La programmation est une opération indispensable avant d'utiliser les EPROM, malheureusement celle-ci met en œuvre un équipement spécial composé d'un programmateur d'EPROM et d'un effaceur aux ultra-violets. Il n'existe pas dans le commerce d'équipement de ce genre compatible avec le ZX 81.

Sa réalisation n'est pourtant pas très compliquée, mais elle dépasse pour l'instant le cadre de cet ouvrage.

Cependant pour les quelques lecteurs désireux de se lancer dans un tel projet de réalisation d'un programmateur d'EPROM , et pour tout le monde, à des fins didactiques, il peut être intéressant de connaître le processus de programmation des 2716.

Initialement, après chaque effacement, obtenu en exposant le boîtier à une dose d'ultra-violets suffisante, tous les bits de la 2716 sont positionnés à l'état logique "1". La programmation des données consiste donc à introduire des "0" dans les cases mémoires binaires désirées.



Pendant cette opération une tension continue de 25 V doit être maintenue sur la broche VPP (broche 21), et lorsque la donnée à programmer ainsi que l'adresse sélectionnée sont présentées sur les deux bus externes, on envoie une impulsion positive de programmation, niveau 5 V et durant entre 50 et 55 ms sur la broche CE/ (broche 18). Pendant toute la programmation l'entrée OE/ doit être appliquée au + 5 V.

Dans l'industrie les programmateurs d'EPROM sont construits à l'aide de coupleurs parallèles comme par exemple le PIA dont nous avons déjà parlé. Alors si le cœur vous en dit...

8.1.4. Recyclage des RAM

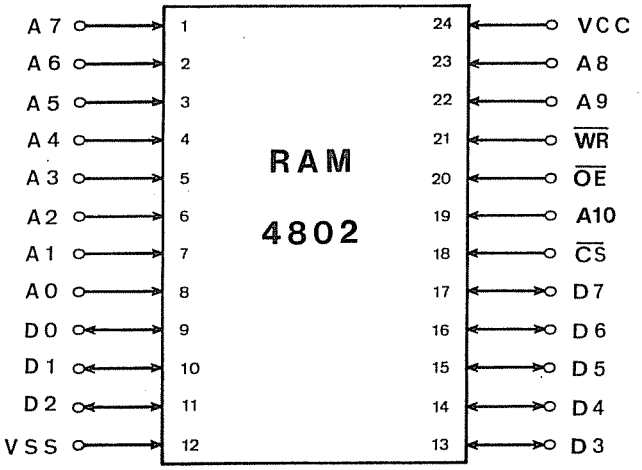
Nous ouvrons ici une parenthèse pour les lecteurs qui en des temps anciens auraient étendu leur version de base à 2K avec une 4802 ou encore à qui il reste la 4118 de base ; en effet sachez que les brochages de ces RAM sont quasiment compatibles avec celui de l'EPROM 2716. Comparez vous-mêmes :

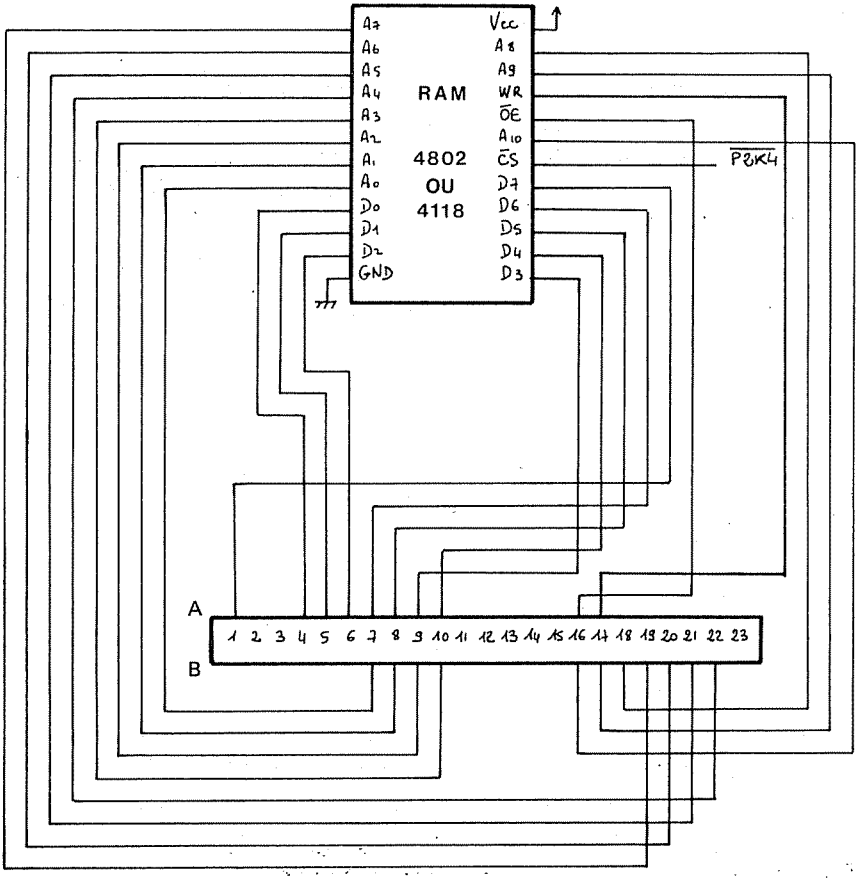
Les différents brochages sont donnés au court de ce livre et vous constaterez que les différences se situent sur les broches 19 et 21 conformément au tableau suivant :

	Broches	
	19	21
EPROM 2716	A10	VPP
RAM 1K 4118	NC	WR/
RAM 2K 4802	A10	WR/

NC signifie non connecté.

Vous voyez alors très bien qu'un minimum de modifications sur le schéma précédant permettra d'installer à la place de l'EPROM indifféremment une RAM 4118 (1K) ou une RAM 4802 (2K) et ainsi de disposer d'une zone de mémoire vive non contrôlée par le BASIC du ZX 81.





8.1.5. Nomenclature des composants utilisés

Nous donnons sous la forme d'un tableau la nomenclature complète des composants utilisés dans cette extension :

Nombre	*Nomenclature*	*Description*	*Commentaire*
1	2716	EPROM	
1	4802	RAM 2K octets	Éventuellement
1	4118	RAM 1K octets	Éventuellement
1	Support 24 broches	A wrapper	

9

RESET manuel

Un petit inconvénient du ZX 81 est que, lorsque le micro-ordinateur se plante, c'est-à-dire que l'utilisateur perd complètement le contrôle de sa machine, le seul moyen utilisable pour avoir à nouveau la possibilité d'utiliser son système est de couper carrément l'alimentation puis de la remettre. Ce procédé réalise indirectement une réinitialisation du CPU qui recommence à dérouler son programme résident à partir de l'adresse 0000H, on l'appelle un RESET automatique.

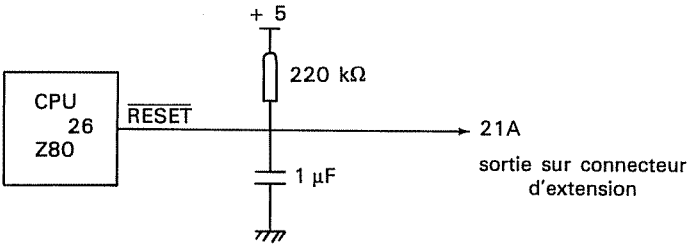
Les inconvénients majeurs d'une telle action, sont d'une part: les coupures d'alimentation répétées finissent à la longue par abimer certains composants électroniques ce qui diminue la durée de vie de votre ZX 81, d'autre part et c'est plus grave encore: les mémoires vives voient leurs contenus détruits lorsqu'on arrête de les alimenter, ce qui est dans certains cas bien gênant. Il serait donc d'un très grand intérêt de disposer sur notre système d'une possibilité de provoquer ce RESET, autrement dit d'avoir un RESET manuel.

9.1. PRINCIPE DE FONCTIONNEMENT DU RESET AUTOMATIQUE DU ZX 81

Lorsque nous reprenons le plan d'ensemble du ZX 81 et que nous recherchons la broche RESET du CPU (broche 26), nous constatons que

celui-ci est connecté à une résistance de 22 kΩ (R15) reliée au + 5 V et à un condensateur (C5) relié à la masse électrique (0 V), conformément au schéma ci-dessous.

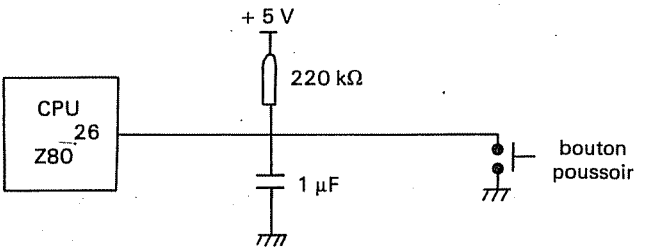
RESET automatique du ZX 81



Le fonctionnement d'un tel ensemble ou cellule RC pour résistance-capacité est classique en électronique : à la mise sous-tension du ZX 81, le + 5 V est appliqué sur R15 dans laquelle va s'établir progressivement un courant pour charger la capacité C5. De cette manière le point milieu directement relié au RESET/ passe en douceur de + 5 V à 0 V ce qui provoque finalement l'initialisation automatique du processeur.

9.2. UNE PETITE AMÉLIORATION

Elle est très simple et très vite réalisée. Son principe est d'installer entre le RESET/ qui sort sur le connecteur d'extension et la masse électrique (0 V) un bouton poussoir comme sur le schéma ci-après.



Le bouton poussoir, lorsqu'il est relâché ne perturbe en aucun cas le fonctionnement du ZX 81, puisqu'il se comporte alors comme un cir-

cuit ouvert. Par contre, une petite pression sur celui-ci met d'un seul coup le condensateur C5 en court-circuit; il se décharge et la broche 26 du CPU voit alors 0 V, c'est son niveau actif et la réinitialisation du CPU est effectuée. Ce montage permet de plus par rapport au RESET automatique de sauvegarder le contenu des RAM statiques qui ne sont pas sous contrôle du BASIC, c'est-à-dire en-dehors des adresses comprises entre 16 et 32K. C'est par exemple le cas si vous avez réalisé l'extension précédente avec soit une RAM 4802 soit encore la 4118.

Annexe 1

La numération en base quelconque

Il est possible de définir d'autres systèmes numériques que le système décimal que nous utilisons couramment et dans lequel le multiple de l'unité est la dizaine. Dans ce dernier cas on dit qu'on a à faire à un système en base 10. En effet, par définition, la base est le nombre choisi comme multiple immédiat de l'unité.

Notons B la base du système qu'on désire utiliser ; dans ce cas on définira :

$B^1 = B$ (énoncé : B puissance 1)

$B^2 = B * B$

$B^3 = B^2 * B = B * B * B$

$B^4 = B^3 * B = B * B * B * B$ etc...

$B^0 = 1$ par définition

L'ÉCRITURE EN BASE QUELCONQUE

On adoptera désormais la notation indicielle suivante : un nombre écrit dans une base quelconque est formé d'une chaîne de chiffres tous inférieurs à la base, juxtaposés puis indicée par la base. Exemples :

1001110_{2}	en base deux ou binaire
7652_{8}	en base huit ou octal
6809_{10}	en base dix ou décimal

par contre une notation de la forme : 11010301 est erronée en base 2 puisque un chiffre composant la chaîne est supérieur à la base d'écriture, mais est tout à fait bonne en base 4 par exemple.

La signification d'une telle notation est la suivante :

$$PQRS_{B} = P * B^{3} + Q * B^{2} + R * B^{1} + S * B^{0}$$

des exemples rendront la compréhension plus rapide :

$$1010_{2} = 1 * 2^{3} + 0 * 2^{2} + 1 * 2^{1} + 0 * 2^{0}$$

$$= 1 * 8^{1} + 0 * 4 + 1 * 2 + 0 * 1 = 10 \text{ ou } 10_{10}$$

$$56_{8} = 5 * 8^{1} + 6 * 8^{0} = 5 * 8 + 5 * 1 = 45 \text{ ou } 45_{10}$$

Pour les bases supérieures à dix il ne suffisait plus des seuls 10 nombres décimaux habituels alors on a utilisé les lettres de l'alphabet par exemple, pour former un nombre en base 16 on utilise les caractères :

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E et F

et on a affecté les valeurs décimales suivantes aux caractères spéciaux :

A	valeur	10
B	valeur	11
C	valeur	12
D	valeur	13
E	valeur	14
F	valeur	15

Ainsi :

$$FE9_{16} = F * 16^2 + E * 16^1 + 9 * 16^0$$
$$= 15 * 256 + 14 * 16 + 9 * 1 = 4073_{10}$$

Les bases les plus couramment utilisées sont :

— le binaire ou base 2,
— l'octal ou base huit,
— le décimal ou base dix,
— l'hexadécimal ou base 16.

Notons encore que l'on rencontre aussi fréquemment les notations suivantes :

suffixe B pour les nombres écrits en binaire par exemple : 1001B
suffixe H pour les nombres écrits en hexadécimal par exemple : C3F5H

LE CALCUL BINAIRE

Nous venons de voir qu'un nombre en binaire s'écrit uniquement avec des "0" et "1", ces deux chiffres s'appellent alors des bits. Un exemple de 8 bits s'appelle un octet. A partir de là, on définit plusieurs opérations logiques et arithmétiques sur les bits. Il est aussi très aisé d'utiliser pour représenter les opérations des tables de vérité ; celle-ci comprennent une colonne par opérande et une colonne supplémentaire pour le résultat de l'opération. Comme un opérande binaire ne peut prendre que les deux valeurs 0 ou 1 on peut faire figurer, en extension, tous les cas possibles rencontrés dans les opérations.

Lorsqu'on applique plus particulièrement le calcul binaire à la représentation des états logiques possibles sur une ligne de circuit électronique, on utilise pour chacune des opérations qui vont suivre des symboles électriques appelés portes logiques spécifiques de l'opération.

Une porte logique comprend des lignes qui entrent et une ligne qui sort, au passage de la porte les signaux entrant sont composés dans l'opération rappelée par la porte pour former le signal sortant. Nous donnons dans la suite les symboles électriques correspondant aux opérations.

Les opérations logiques

La complémentation ou inversion

La complémentation consiste à inverser la valeur du bit: zéro devient un et un devient zéro.

A	A
0	1
1	0

porte inverseuse

La conjonction: "et logique" ("AND" en anglais)

Le "et logique" de deux chiffres binaires vaut 1 si les deux chiffres sont A 1 et 0 dans tous les autres cas.

ce qui donne la table de vérité suivante:

A	B	A. B
0	0	0
0	1	0
1	0	0
1	1	1

porte AND

La disjonction: "ou logique" ("OR" en anglais)

Le "ou logique" de deux chiffres binaires vaut 1 si l'un des deux au moins vaut 1 et 0 si tous les deux sont nuls.

Ce qui donne la table de vérité suivante:

A	B	A + B	
0	0	0	porte OR
0	1	1	
1	0	1	
1	1	1	

La disjonction exclusive: "ou exclusif logique" ("XOR" en anglais)

Le "ou exclusif logique" de deux chiffres binaires vaut 1 si l'un des deux chiffres et un seul vaut 1 et 0 si tous les deux sont nuls ou si tous les deux valent 1.

Ce qui donne la table de vérité suivante:

A	B	A * B	
0	0	0	porte XOR
0	1	1	
1	0	1	
1	1	0	

Les théorèmes de Morgan

Les deux théorèmes qui suivent montrent comment on peut transformer certaines opérations logiques.

— Le complément d'une disjonction est la conjonction des compléments.

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

soit encore avec des portes logiques.

— Le complément d'une conjonction est la disjonction des compléments.

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

soit encore avec des portes logiques.

LE CALCUL EN BASE 16 OU CALCUL HEXADÉCIMAL

Nous venons de voir que la représentation binaire convenait bien à la représentation des niveaux logiques qu'utilisent les microprocesseurs. Seulement il est très difficile de manipuler des trains de bits sans accomplir des erreurs ; c'est pourquoi on a pensé à regrouper les bits quatre par quatre et ainsi à former des demi-octets. Le plus grand nombre que l'on puisse former avec quatre octets est :

$$1111_{2} = 15_{10}$$

Ce qui a fait émerger l'idée qu'on pouvait utiliser le système en base 16 ou hexadécimal. De cette manière la correspondance entre demi-octets et nombre hexadécimal est la suivante :

Nombre binaire		*Nombre hexadécimal*
0000	=	0
0001	=	1
0010	=	2
0011	=	3
0100	=	4
0101	=	5
0110	=	6
0111	=	7
1000	=	8
1001	=	9
1010	=	A
1011	=	B
1100	=	C
1101	=	D
1110	=	E
1111	=	F

Un exemple de cette règle de compactage peut être :

Chaîne binaire :	0011	0101	0010	1110	0011	1010	1100
Chaîne hexadécimale équivalente :	3	5	2	E	3	A	C

L'encombrement des listings est alors d'un seul coup divisé par quatre et il s'avère que ce type d'écriture permet de réduire le nombre des erreurs de lecture.

Les opérations entre les bits définies plus haut s'étendent aussi aux chaînes de bits puis aux nombres hexadécimaux.

Ainsi par exemple :

3EH . 6BH = 2AH

car c'est la représentation de l'opération binaire :

0011 1110 B . 0110 1011 B = 0010 1010 B

Annexe 2

Instructions du Z 80

Tableau des codes « opération »

00	NOP		17	RLA	
01	LD	BC,+dddd	18	JR	d
02	LD	(BC),A	19	ADD	HL,DE
03	INC BC		1A	LD	A,(DE)
04	INC B		1B	DEC DE	
05	DEC B		1C	INC E	
06	LD	B,+dd	1D	DEC E	
07	RLCA		1E	LD	E,+dd
08	EX	AF,A'F'	1F	RRA	
09	ADD	HL,BC	20	JR	NZ,d
0A	LD	A,(BC)	21	LD	HL,+dddd
0B	DEC	BC	22	LD	(addr.) HL
0C	INC C		23	INC HL	
0D	DEC C		24	INC H	
0E'	LD	C,+dd	25	DEC H	
0F	RRCA		26	LD	H,+dd
10	DJNZ	d	27	DAA	
11	LD	DE,+dddd	28	JR	Z,d
12	LD	(DE),A	29	ADD	HL,HL
13	INC DE		2A	LD	HL,(addr.)
14	INC D		2B	DEC HL	
15	DEC D		2C	INC L	
16	LD	D,+dd	2D	DEC L	

2E	LD	L,+dd
2F	CPL	
30	JR	NC,d
31	LD	SP,+dddd
32	LD	(addr.), A
33	INC SP	
34	INC	(HL)
35	DEC	(HL)
36	LD	(HL),+dd
37	SCF	
38	JR	C,d
39	ADD	HL,SP
3A	LD	A,(addr.)
3B	DEC SP	
3C	INC A	
3D	DEC A	
3E	LD	A,+dd
3F	CCF	
40	LD	B,B
41	LD	B,C
42	LD	B,D
43	LD	B,E
44	LD	B,H
45	LD	B,L
46	LD	B,(HL)
47	LD	B,A
48	LD	C,B
49	LD	C,C
4A	LD	C,D
4B	LD	C,E
4C	LD	C,H
4D	LD	C,L
4E	LD	C,(HL)
4F	LD	C,A
50	LD	D,B
51	LD	D,C
52	LD	D,D
53	LD	D,E
54	LD	D,H
55	LD	D,L
56	LD	D,(HL)
57	LD	D,A
58	LD	E,B
59	LD	E,C
5A	LD	E,D
5B	LD	E,E
5C	LD	E,H
5D	LD	E,L
5E	LD	E,(HL)
5F	LD	E,A
60	LD	H,B
61	LD	H,D
62	LD	H,C
63	LD	H,E
64	LD	H,H
65	LD	H,L
66	LD	H,(HL)
67	LD	H,A
68	LD	L,B
69	LD	L,C
6A	LD	L,D
6B	LD	L,E
6C	LD	L,H
6D	LD	L,L
6E	LD	L,(HL)
6F	LD	L,A
70	LD	(HL),B
71	LD	(HL),C
72	LD	(HL),D
73	LD	(HL),E
74	LD	(HL),H
75	LD	(HL),L
76	HALT	
77	LD	(HL),A
78	LD	A,B
79	LD	A,C
7A	LD	A,D
7B	LD	A,E
7C	LD	A,H
7D	LD	A,L
7E	LD	A,(HL)
7F	LD	A,A
80	ADD	A,B
81	ADD	A,C

82	ADD	A,D
83	ADD	A,E
84	ADD	A,H
85	ADD	A,L
86	ADD	A,(HL)
87	ADD	A,A
88	ADC	A,B
89	ADC	A,C
8A	ADC	A,D
8B	ADC	A,E
8C	ADC	A,H
8D	ADC	A,L
8E	ADC	A,(HL)
8F	ADC	A,A
90	SUB	B
91	SUB	C
92	SUB	D
93	SUB	E
94	SUB	H
95	SUB	L
96	SUB	(HL)
97	SUB	A
98	SBC	A,B
99	SBC	A,C
9A	SBC	A,D
9B	SBC	A,E
9C	SBC	A,H
9D	SBC	A,L
9E	SBC	A,(HL)
9F	SBC	A,A
A0	AND	B
A1	AND	C
A2	AND	D
A3	AND	E
A4	AND	H
A5	AND	L
A6	AND	(HL)
A7	AND	A
A8	XOR	B
A9	XOR	C
AA	XOR	D
AB	XOR	E
AC	XOR	H
AD	XOR	L
AE	XOR	(HL)
AF	XOR	A
B0	OR	B
B1	OR	C
B2	OR	D
B3	OR	E
B4	OR	H
B5	OR	L
B6	OR	(HL)
B7	OR	A
B8	CP	B
B9	CP	C
BA	CP	D
BB	CP	E
BC	CP	H
BD	CP	L
BE	CP	(HL)
BF	CP	A
C0	RET	NZ
C1	POP	BC
C2	JP	NZ,addr.
C3	JP	addr.
C4	CALL	NZ,addr.
C5	PUSH	BC
C6	ADD	A,+dd
C7	RST	00
C8	RET	Z
C9	RET	
CA	JP	A,addr.
CB		
CC	CALL	Z,addr.
CD	CALL	addr.
CE	ADC	A,+dd
CF	RST	08 H
D0	RET	NC
D1	POP	DE
D2	JP	NC,addr.
D3	OUT	(+dd)A
D4	CALL	NC, addr.
D5	PUSH	DE

D6	SUB	+dd
D7	RST	10 H
D8	RET C	
D9	EXX	
DA	JP	C,addr.
DB	IN	A,(+dd)
DC	CALL	C,addr.
DD	voir ci-après	
DE	SBC	A,+dd
DF	RST	18 H
E0	RET PO	
E1	POP HL	
E2	JP	PO,addr.
E3	EX	(HL),SP
E4	CALL	PO,addr.
E5	PUSH	HL
E6	AND	+dd
E7	RST	20 H
E8	RET PE	
E9	JP	(HL)
EA	JP	PE,addr.
EB	EX	DE,HL
EC	CALL	PE,addr.
ED	voir ci-après	
EE	XOR	+dd
EF	RST	28 H
F0	RET P	P
F1	POP AF	AF
F2	JP	P,addr.
F3	DI	
F4	CALL	P,addr.
F5	PUSH	AF
F6	OR	+dd
F7	RST	30 H
F8	RET M	
F9	LDS	SP,HL
FA	JP	M,addr.
FB	EI	
FC	CALL	M,addr.
FD	voir ci-après	
FE	CP	+dd
FF	RST	38 H

Instructions ED

ED 40	IN	B,(C)
ED 41	OUT	(C),B
ED 42	SBC	HL,BC
ED 43	LD	(addr.),BC
ED 44	NEG	
ED 45	RETN	
ED 46	IM 0	
ED 47	LD	I,A
ED 48	IN	C,(C)
ED 49	OUT	(C),C
ED 4A	ADC	HL,BC
ED 4B	LD	BC,(addr.)
ED 4D	RETI	
ED 4F	LD	R,A
ED 50	IN	D,(C)
ED 51	OUT	(C),D
ED 52		SBC HL,DE
ED 53	LD	(addr.),DE
ED 56	IM	1
ED 57	LD	A,1
ED 58	IN	E,(C)
ED 59	OUT	(C),E
ED 5A	ADC	HL,DE
ED 5B	LD	DE,(addr.)
ED 5F	LD	A,R
ED 60	IN	H,(C)
ED 61	OUT	(C),H
ED 62	SBC	HL,HL
ED 63	LD	(addr.),HL
ED 66	IM	2
ED 67	RRD	
ED 68	IN	L,(C)

ED 69	OUT	(C),L		ED A3	OUTI
ED 6A	ADC	HL,HL		ED A8	LDD
ED 6B	LD	HL,(addr.)		ED A9	CPD
ED 6F	RLD			ED AA	IND
ED 72	SBC	HL,SP		ED AB	OUTB
ED 73	LD	(addr.),SP		ED B0	LDIR
ED 78	IN	A,(C)		ED B1	CPIR
ED 79	OUT	(C),A		ED B2	INIR
ED 7A	ADC	HL,SP		ED B8	LDDR
ED 7B	LD	SP,(addr.)		ED B9	CPDR
ED A0	LDI			ED BA	INDR
ED A1	CPI			ED BB	OTRD
ED A2	INI				

Instructions DD et FD

Nous ne donnons que les codes des instructions DD qui travaillent sur le registre d'index IX, les instructions travaillant sur le registre d'index IY sont les mêmes avec la différence suivante. Leurs codes commencent par FD au lieu de DD.

DD 09		ADD	IX,BC	DD 71	d	LD	(IX+d),C
DD 19		ADD	IX,DE	DD 72	d	LD	(IX+d),D
DD 21	+dddd	LD	IX,+dddd	DD 73	d	LD	(IX+d),E
DD 22	addr.	LD	(addr.),IX	DD 74	d	LD	(IX+d),H
DD 23		INC	IX	DD 75	d	LD	(IX+d),L
DD 29		ADD	IX,IX	DD 77	d	LD	(IX+d),A
DD 2A	addr.	LD	IX,(addr.)	DD 7E	d	LD	A,(IX+d)
DD 2B		DEC	IX	DD 86	d	ADD	A,(IX+d)
DD 34	d	INC	(IX+d)	DD 8E	d	ADC	A,(IX+d)
DD 35	d	DEC	(IX+d)	DD 96	d	SUB	(IX+d)
DD 36	d+dd	LD	(IX+d),+dd	DD 9E	d	SBC	A,(IX+d)
DD 39		ADD	IX,SP	DD A6	d	AND	(IX+d)
DD 46	d	LD	B,(IX+d)	DD AE	d	XOR	(IX+d)
DD 4E	d	LD	C,(IX+d)	DD B6	d	OR	(IX+d)
DD 56	d	LD	D,(IX+d)	DD BE	d	CP	(IX+d)
DD 5E	d	LD	E,(IX+d)	DD CB	d 06	RLC	(IX+d)
DD 66	d	LD	H,(IX+d)	DD CB	d 0E	RRC	(IX+d)
DD 6E	d	LD	L,(IX+d)	DD CB	d 16	RL	(IX+d)
DD 70	d	LD	(IX+d),B	DD CB	d 1E	RR	(IX+d)

DD CB d	26	SLA	(IX+d)
DD CB d	2E	SRA	(IX+d)
DD CB d	3E	SRL	(IX+d)
DD CB d	46	BIT	0,(IX+d)
DD CB d	4E	BIT	1,(IX+d)
DD CB d	56	BIT	2,(IX+d)
DD CB d	5E	BIT	3,(IX+d)
DD CB d	66	BIT	4,(IX+d)
DD CB d	6E	BIT	5,(IX+d)
DD CB d	76	BIT	6,(IX+d)
DD CB d	7E	BIT	7,(IX+d)
DD CB d	86	RES	0,(IX+d)
DD CB d	8E	RES	1,(IX+d)
DD CB d	96	RES	2,(IX+d)
DD CB d	9E	RES	3,(IX+d)
DD CB d	A6	RES	4,(IX+d)
DD CB d	AE	RES	5,(IX+d)
DD CB d	B6	RES	6,(IX+d)
DD CB d	BE	RES	7,(IX+d)
DD CB d	C6	SET	0,(IX+d)
DD CB d	CE	SET	1,(IX+d)
DD CB d	D6	SET	2,(IX+d)
DD CB d	DE	SET	3,(IX+d)
DD CB d	E6	SET	4,(IX+d)
DD CB d	EE	SET	5,(IX+d)
DD CB d	F6	SET	6,(IX+d)
DD CB d	FE	SET	7,(IX+d)
DD E1		POP	IX
DD E3		EX	(SP),IX
DD E5		PUSH	IX
DD E9		JP	(IX)
DD F9		LD	SP,IX

Annexe 3

Instructions Z 80 travaillant sur des mémoires ou des entrées-sorties

Avec les mémoires		Avec les entrées-sorties	
	CHARGEMENT		
	Adressage indirect		
LD	A, (NN)	IN	A, (N)
	Adressage indirect par registre		
LD	A, (BC)	IN	R, (C)
LD	A, (DE)		
LD	A, (HL)		
	Incrémentation automatique		
LDI		INI	
	Incrémentation et répétition automatiques		
LDIR		INIR	
	Décrémentation automatique		
LDD		IND	
	Décrémentation et répétition automatiques		
LDDR		INDR	

STOCKAGE

Adressage indirect

LD	(NN), A	OUT	(N), A

Adressage indirect par registre

LD	(BC), A	OUT	(C), R
LD	(DE), A		
LD	(HL), A		

Incrémentation automatique

OUTI

Incrémentation et répétition automatiques

OTIR

Décrémentation automatique

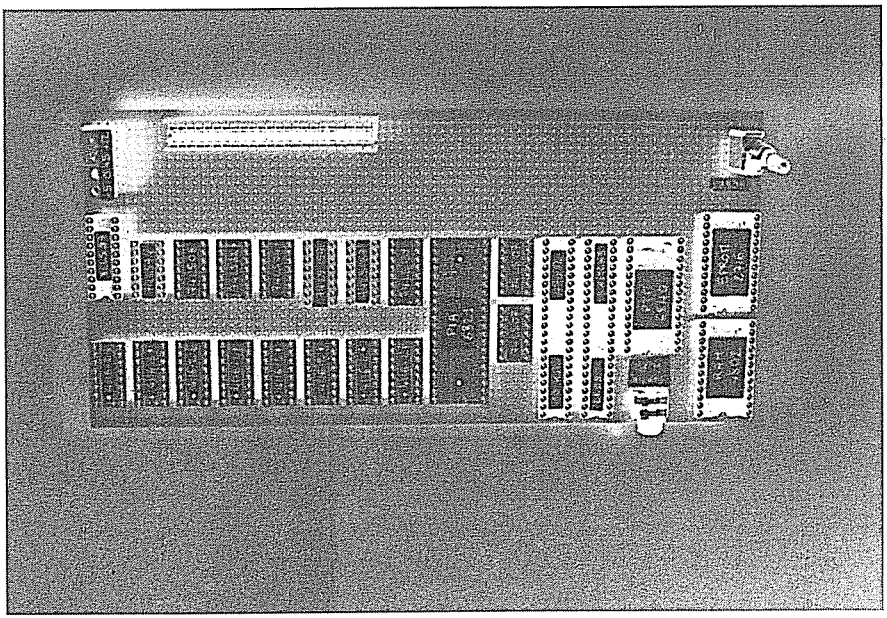
OUTD

Décrémentation et répétition automatiques

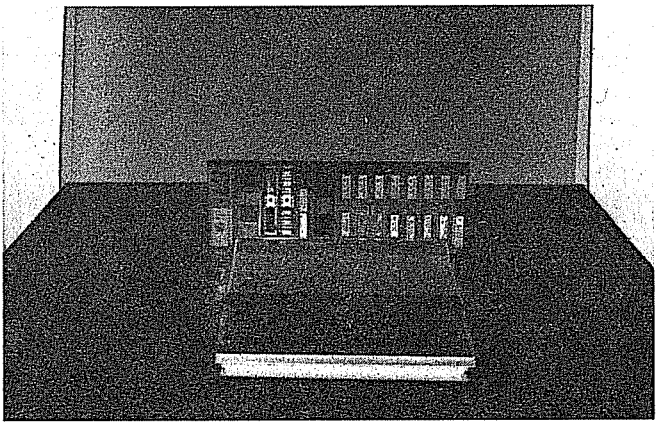
OTDR

Annexe 4

Implantation d'ensemble des extensions



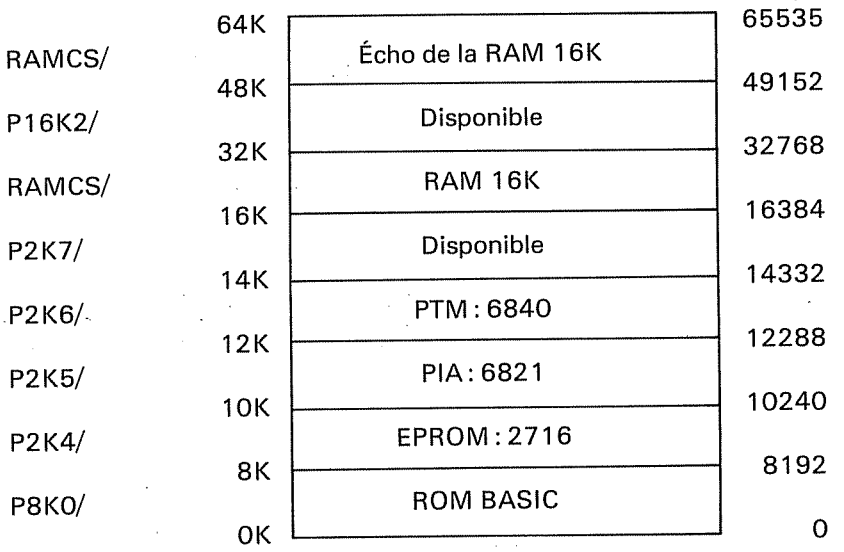
Implantation des composants sur la carte extension.



Le ZX 81 et sa carte d'extension.

Annexe 5

Carte mémoire du ZX 81 avec les extensions de ce livre



Annexe 6

Liste non exhaustive de fournisseurs pouvant approvisionner le matériel cité

Nom	*Adresse*	*Composants micro*	*Matériel wrapping*
Radio-son	31, rue Néricault-Destouches 37000 Tours Tél. : (47) 20.80.19	X	X
Microprоss	79, Av. du Gal de Gaulle 68000 Colmar Tél. : (89) 23.25.11	X	
Béric	43, rue Victor Hugo 92240 Malakoff Tél. : 657.68.33	X	X
Penta 8	34, rue de Turin 75008 Paris Tél. : 293.41.33	X	X
Penta 13	10, Bd Arago 75013 Paris Tél. : 336.26.05	X	X
Penta 16	5, rue M. Bourdet 75016 Paris Tél. : 524.23.16	X	X

Albion	9, rue de Budapest 75009 Paris Tél. 874.14.14	X	
Cirque Radio	24, Bd des Filles du Calvaire 75011 Paris Tél. : 805.22.76	X	
Radio Prim	5, rue de l'Aqueduc 75010 Paris Tél. : 607.05.15	X	
Acer composants	42, rue de Chabrol 75010 Paris Tél. : 770.28.31	X	
Reuilly composants	79, Bd Diderot 75012 Paris Tél. : 372.70.17	X	
Montparnasse composants	3, rue du Maine 75014 Paris Tél. 320.37.10	X	
Soamet S.A. (importateur OK)	10, Bd F. Hostachy 78290 Croissy/Seine Tél. : 976.24.37		X

Liste des illustrations

Brochages

Chronogrammes

Schémas électriques

Schémas divers

Photographies

Imprimé en France. — JOUVE, 18, rue Saint-Denis, 75001 PARIS
N° 11613. Dépôt légal : Juillet 1983
N° d'Editeur : 3937

A tous ceux qui ont vécu leur première expérience informatique avec le ZX 81, et qui pensent désormais avoir fait le tour de ce petit système :

Détrompez-vous !... Et ne mettez pas encore votre ordinateur au rebut : « DES EXTENSIONS A CONSTRUIRE POUR VOTRE ZX 81 » va vous permettre d'appréhender une dimension d'utilisation tout à fait nouvelle !

En effet, notre propos principal ne sera pas le logiciel, au même titre que la plupart des ouvrages publiés, mais le matériel : vous allez acquérir l'approche de l'électronicien, du concepteur de matériel !

Cet apprentissage est très progressif. Il débute par un chapitre sur les systèmes à microprocesseur, votre ZX 81 étant évidemment là pour imager le récit.

Puis, une fois que vous aurez bien assimilé les bases, nous nous attaquerons à des réalisations pratiques d'extensions pour le ZX 81... Et le terme « réalisation » n'est pas trop fort, car tout y est : la technique, la méthode et les explications complètes, afin que vous puissiez parvenir dans les meilleures conditions au montage d'extensions aussi variées que :

- Un coupleur parallèle permettant le dialogue avec l'environnement.
- Une extension mémoire dynamique 16 K puis 32 K.
- Un générateur de notes.
- Une extension mémoire morte destinée à recevoir un supplément de logiciel résident...

EYROLLES

8638