

Denis BONOMO

Eddy DUTERTRE

COMMUNIQUEZ AVEC VOTRE ZX 81 OU AVEC VOTRE TIMEX 1000



SORACOM
editions
INFORMATIQUE



**COMMUNIQUEZ
AVEC VOTRE
ZX 81**



Denis BONOMO
F6GKQ

Eddy DUTERTRE
FIEZH

COMMUNIQUEZ
AVEC VOTRE
ZX 81

DIFFUSION:

EDITIONS SORACOM

16 A, avenue Gros Malhon- BP 5075
35025 RENNES cedex

«La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part que «les copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective» et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, «toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants-droit ou ayants-cause, est illicite» (alinéa premier de l'article 40). Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles 425 du Code Pénal.»

© Éditions SORACOM – 1983
ISBN 2-904032-06-1

Aux Editions Soracom

OUVRAGES PARUS

TECHNIQUE RADIO POUR L'AMATEUR
de F. Mellet et S. Faurez

LA GUERRE DES ONDES
de F. Mellet et S. Faurez

A L'ÉCOUTE DES RADIOTÉLÉTYPES
de J.L. Fis

ALIMENTATIONS DE PUISSANCE
collection «Sélection de Montages»

INTERFÉRENCES TV (QRM TV)
de F. Mellet et K. Pierrat

LES QSO EN RADIOTÉLÉPHONIE (Français-Anglais)
de L. Sigrand

TRANSAT TERRE-LUNE
de l'Union pour la Promotion de la Propulsion Photonique

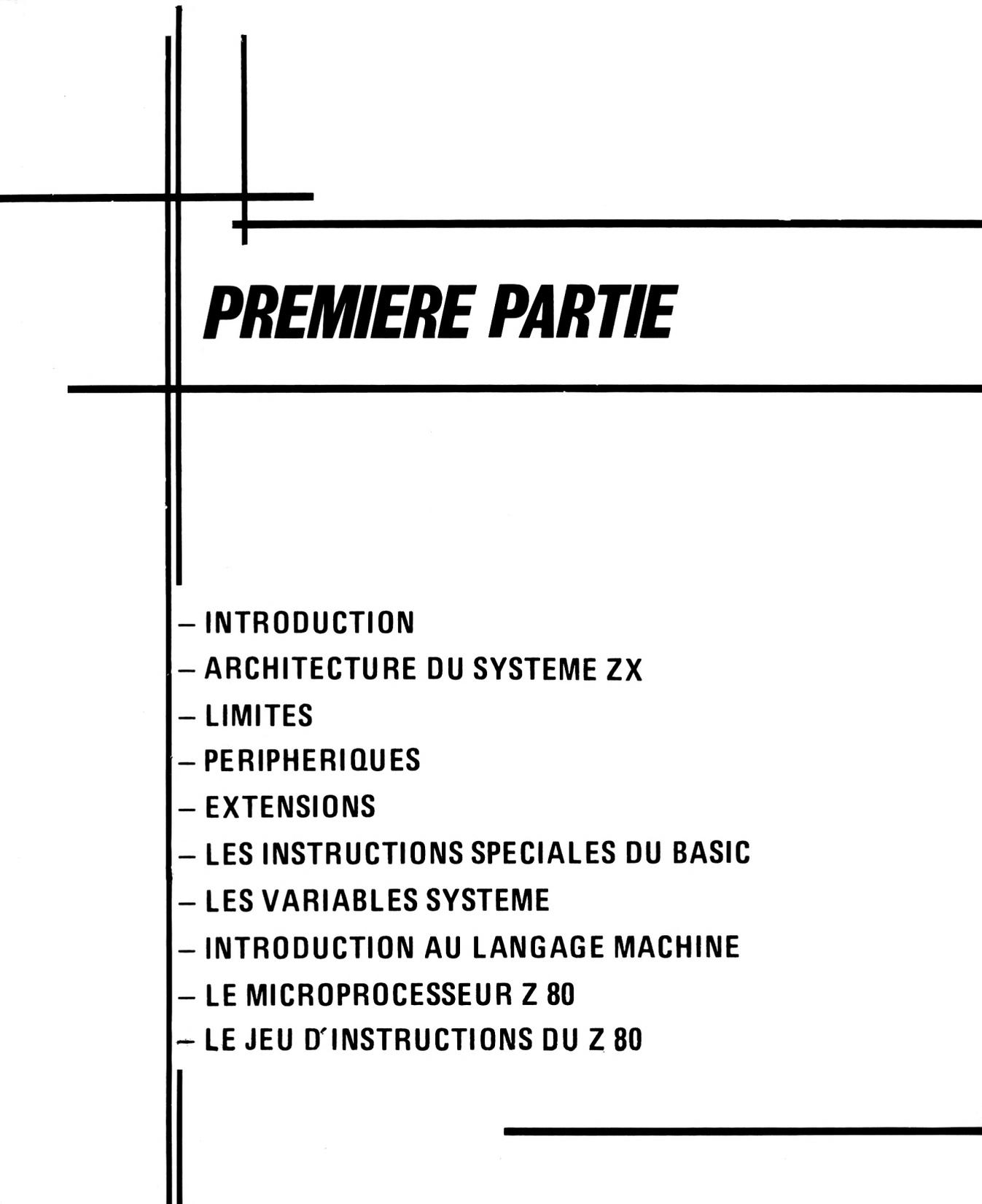
OUVRAGES EN PRÉPARATION

LA RÉCEPTION DES SATELLITES MÉTÉO
de L. Kuhlmann

TÉLÉVISION LONGUE DISTANCE
de P. Godou

ASTRONOMIE D'AMATEUR
de Th. Lombry

TECHNIQUE DE LA B.L.U.
de G. Ricaud (2ème édition)



PREMIERE PARTIE

- INTRODUCTION
- ARCHITECTURE DU SYSTEME ZX
- LIMITES
- PERIPHERIQUES
- EXTENSIONS
- LES INSTRUCTIONS SPECIALES DU BASIC
- LES VARIABLES SYSTEME
- INTRODUCTION AU LANGAGE MACHINE
- LE MICROPROCESSEUR Z 80
- LE JEU D'INSTRUCTIONS DU Z 80

INTRODUCTION

L'informatique envahit notre vie à tous les niveaux. Si certains systèmes informatisés apparaissent dans les administrations comme pénalisants pour les usagers parce qu'ils ne sont pas ou mal adaptés, d'autres rendent bien des services et raccourcissent considérablement des tâches devenues quotidiennes. Les gros systèmes ne nous sont pas familiers, mais les micro-systèmes, eux, sont apparus un petit peu partout et excitent notre curiosité.

Le micro-processeur à l'origine de cette prolifération se retrouve au cœur du système de traitement de textes qui équipe votre secrétariat, dans la cuisinière électrique, où il déterminera avec précision les temps de cuisson, dans les instruments de navigation équipant AIRBUS, ou au cœur du banc de test automatique qui a vérifié, en sortie de chaîne, votre dernier transceiver.

Transceiver ? Au fait, mais qui gère le clavier de mise en mémoire des fréquences, l'affichage numérique et autres raffinements ? C'est encore lui !

Il s'avère donc indispensable d'acquérir quelques connaissances, même sommaires, dans ce domaine. Les intérêts de la micro-informatique au service des radioamateurs sont multiples. Le principal n'est-il pas que notre curiosité, sans cesse sollicitée, puisse se trouver satisfaite en appliquant des techniques nouvelles à la radio, technique ancienne en fin de siècle.

Nous avons vu, en effet, que les micro-processeurs sont présents dans nos matériels amateurs, du transceiver au satellite en passant par la logique gérant le relais local. La micro-informatique est aussi un nouveau pôle d'attraction permettant de relancer des activités et des sujets de conversations (autres que la position de l'antenne mobile sur la malle arrière ou sur la gouttière), sur nos bandes amateurs. C'est enfin la possibilité de «bidouiller» sans risque de casser du matériel en utilisant sa matière grise.

Nous lui trouverons donc de multiples applications :

- FICHIERS : Confirmation de pays et départements, suivi de contests, gestion des QSL.
- CALCULS : Passage de satellites et poursuite, calcul des distances par QTH-Locator, position de la lune pour trafic EME.
- TRAFIC : Mires A.T.V. et S.S.T.V., émission-réception CW et RTTY, transmission automatique de données (exemple : DL1WX).

Pour accéder à ce vaste champ d'exploitation, les outils ne manquent pas, et notre choix s'est porté sur une machine qui, malgré son prix réduit, permet de tous les aborder : le ZX 81.

Le but visé par cet ouvrage n'est pas de faire un cours de BASIC ou de programmation, mais de montrer, avec programmes à l'appui, comment tirer au mieux parti du ZX 81 quand on est radioamateur.

Nous essayerons de vous conduire à travers les méandres de la programmation, vers une utilisation rationnelle de votre matériel. Pour ce faire, nous ne reprendrons pas toutes les instructions du BASIC ZX une par une, mais celles qui appellent le plus de commentaires. Il en sera de même avec le langage machine car nous ne voulons pas rivaliser avec les ouvrages qui ne traitent que de ces sujets.

Nous insistons, par contre, lors de la présentation des programmes d'applications, sur les différents listings en les commentant abondamment pour en démonter les mécanismes. Le moyen essentiel sera donc l'exemple à tous les niveaux et nous espérons ainsi rendre ce livre plus attrayant. Grâce à cela, nous pensons démontrer que la programmation peut aussi être abordée comme un jeu et démystifier le langage machine en vous incitant à mieux le connaître.

Pour la partie «matériel» nous aborderons quelques points précis sur lesquels il nous a paru bon d'insister pour rendre notre petite machine plus performante. Quant aux modifications ou adjonctions, elles ont été expérimentées par nos soins et nous vous les proposons sans risque.

Nous n'évoquerons qu'en quelques mots la rapide invasion du marché par cette machine créant un nouveau créneau de micro-ordinateurs à des prix jusque là réservés aux programmables de poche.

Le ZX 80 avec la ROM 4 K et ses 21 circuits intégrés, vendu à son arrivée en France autour de 1 250 F, avait déjà fait des adeptes quand arriva le ZX 81.

Principales différences entre les deux machines : le BASIC (entier sur ZX 80, étendu sur ZX 81), la gestion d'écran (modes LENT et RAPIDE sur ZX 81) et l'intégration (21 circuits dans le ZX 80, 4 dans le ZX 81). Résultat de cette «haute intégration» : une baisse de prix de la machine en dessous de 1 000 F, prix qui a baissé depuis jusqu'aux environs de 500 F en kit.

Pour obtenir un système exploitable il faut une extension mémoire (16 K pour 400 F environ) et, éventuellement, une imprimante. Celle qui est proposée par SINCLAIR pour la «recopie d'écran» est en fait bien suffisante pour une utilisation intermittente.

Avec moins de 2 000 F on dispose ainsi d'un système très complet. Evidemment, il ne peut prétendre rivaliser avec des systèmes plus puissants, mais son prix en fait un outil abordable pour qui veut apprendre à programmer, tant en BASIC qu'en langage machine. Sur ce dernier point, il faut dire que nous aurons quelques restrictions et inconvénients que nous examinerons et qu'il est aisé de comprendre après avoir pris connaissance de l'architecture de la machine.

Outre son prix, le ZX 81 offre d'autres intérêts qui sont :

- une taille réduite permettant un transport aisé,
- la connexion aisée à n'importe quel TV UHF,
- interface cassette fiable si le magnétophone l'est,
- un BASIC assez puissant (mais lent) permettant d'accéder aux calculs scientifiques.

ARCHITECTURE DU SYSTEME --- ---

Tout le système ZX est bâti autour du Z 80, micro-processeur 8 bits très répandu ; nous lui consacrerons un petit chapitre pour mieux le connaître.

Auparavant, examinons les autres composants du système car, sans eux, le Z 80 ne pourrait pas faire grand-chose !

Son principal associé est le circuit LSI (Large Scale Integration = « haute intégration »), développé spécialement pour le ZX 81. Ce circuit remplace toute la logique qu'on trouvait dans le ZX 80 et assure des fonctions multiples allant de la gestion d'écran aux entrées-sorties vers le magnétophone, en passant par la gestion du clavier.

L'ensemble serait muet et bien inefficace si un autre composant essentiel ne venait organiser le séquençage des opérations : il s'agit de la ROM (Read Only Memory = « mémoire morte », c'est-à-dire qu'on ne peut que lire) qui contient le langage BASIC.

Elle permet à la fois l'initialisation du système, sa gestion et l'interprétation du langage. Sa taille est de 8 K MOTS (1 MOT = 8 bits = 1 octet ; 1 K mot = 1024 mots = 1024 octets).

Elle contient aussi le générateur des caractères qui permet de faire apparaître sur l'écran les signes, chiffres et lettres que nous connaissons, qui occupe 512 octets.

Nous trouvons aussi 1 K de RAM (Random Access Memory = « mémoire vive », c'est-à-dire qu'on peut lire mais on peut aussi écrire). C'est ici que se logeront programme, variables, système, etc...

Le synoptique (*Figure 1*) nous permettra de mieux comprendre cette architecture.

Il reste, enfin, le « modulateur » qui transforme en U.H.F. le signal « vidéo-composite » issu du circuit spécialisé pour l'envoyer au téléviseur.

Ce modulateur est un auto-oscillateur modulé en amplitude dont la sortie est réglée autour du canal 36. Pour les modèles vendus en France, il est précédé d'un transistor inverseur pour le rendre compatible avec le système français. En effet, il faut savoir que la vidéo est « positive » au sortir du circuit spécialisé et qu'elle est inversée, pour le système anglais, dans le modulateur. Nous insistons sur ce point car nous décrivons plus loin une sortie vidéo directe permettant une amélioration sensible de l'image.

Il est aisé de comprendre que l'inconvénient d'un tel système où tout est placé sous le contrôle quasi unique du micro-processeur, et qui travaille tout en interruptions, est la lenteur d'exécution.

Ceci nous conduit à examiner les limites du ZX.

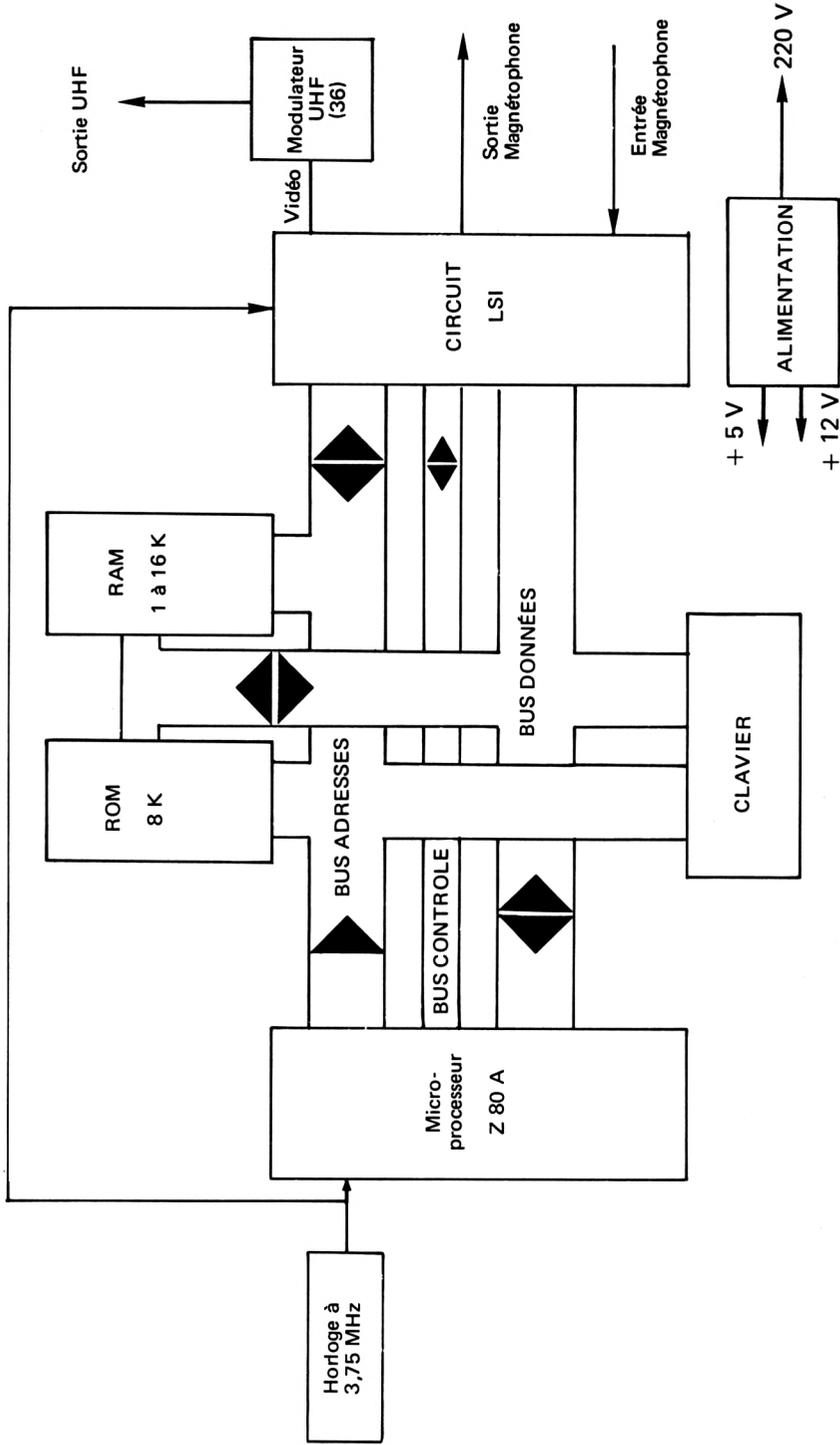
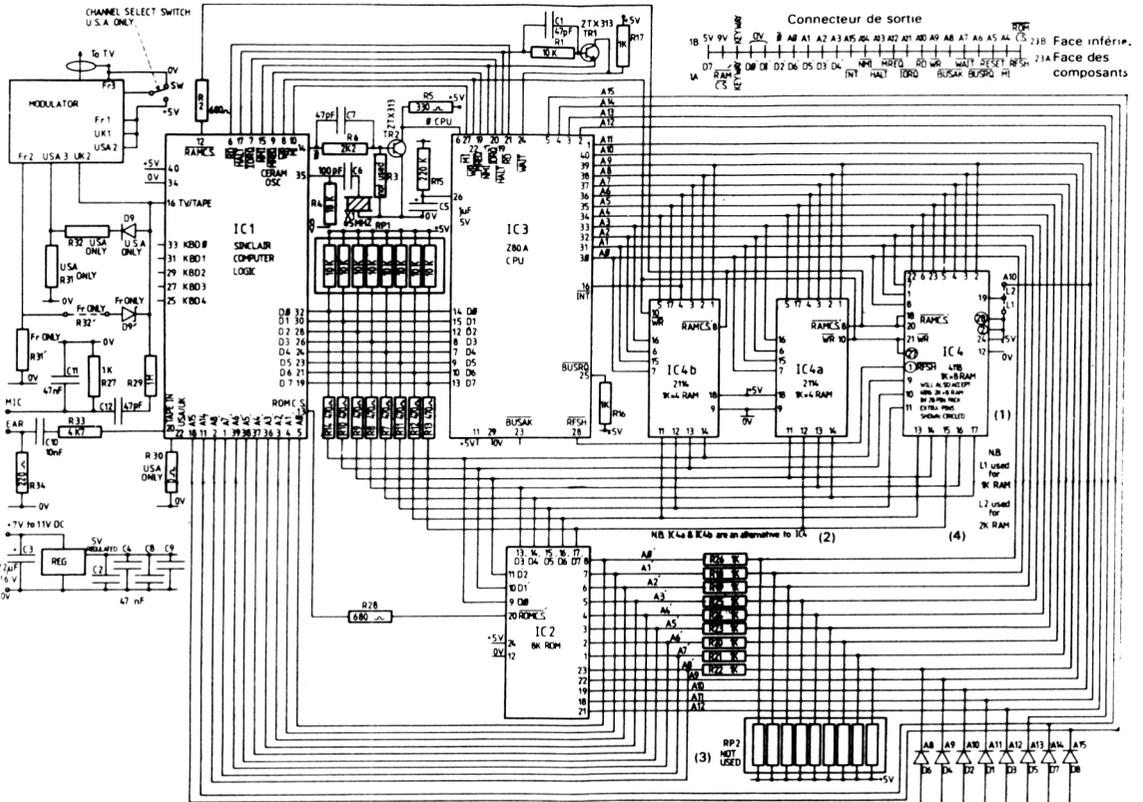


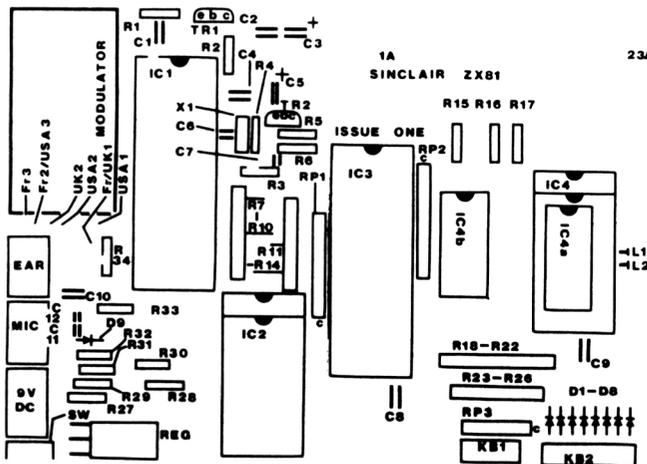
Fig. 1 : SYNOPTIQUE DU ZX 81



(1) Peut aussi être muni d'une RAM 2K X 8 à 28 fiches. (fiches supplémentaires, entourées d'un cercle) (2) NB IC4a & IC4b remplacent parfois IC4 (3) RP2 non utilisé. (4) L1 pour RAM 1 K, L2 pour RAM 2 K

DRW No SRC 049

SINCLAIR ZX81



ZX 81

MS

LIMITES DU ZX

Elles sont surtout dûes à sa conception centralisée autour du Z 80 qui fait pratiquement tout. C'est en grande partie pour cela que le BASIC est lent (et peut-être aussi à cause de la conception du programme en ROM avec de très nombreux déroutements).

Nous verrons que cette lenteur peut être gênante et qu'elle nous conduira, dans certains cas, à travailler en langage machine où, là encore, nous serons parfois limités.

Pour se convaincre du « temps de vacances » imposé au Z 80, on peut mettre un oscilloscope sur sa ligne HALT. On la verra trop souvent active. En gros, on peut dire que, en mode LENT, le Z 80 ne travaille que pendant le temps de blanking ligne et image.

D'autres limites seront contraignantes pour nous, radioamateurs : ce sont les problèmes d'interférences.

INTERFERENCES RADIO – TV

Ces problèmes ont toujours été très importants sur les micro-ordinateurs et le sont tout particulièrement sur le ZX.

La raison principale est que le ZX, entièrement en boîtier plastique, laisse passer dans nos récepteurs les fronts raides de ses signaux logiques, son horloge à 6.5 MHz, les signaux de rafraîchissement mémoire, ceux du « convertisseur statique » de celle-ci, la HF du modulateur vidéo.

Face à ces perturbations, il existe peu de solutions, sinon un blindage efficace destiné à emprisonner ces parasites particulièrement virulents.

Ceci peut être réalisé en enfermant le ZX dans un environnement métallique. Il faut convenir qu'avec le boîtier-clavier d'origine, ce n'est pas facile. En s'armant de patience on peut « tapisser » (par l'intérieur) tout le ZX d'une mince feuille métallique, fixée en plusieurs points à la masse.

L'autre solution, plus efficace, consiste, dans le cas de l'utilisation d'un clavier métallique, à mettre le ZX en boîtier métallique. Pour tirer le meilleur parti de ce blindage, il faudra également y enfermer toutes les extensions (mémoires, entrées-sorties, etc...) et relier la masse générale à une prise de terre efficace. Si on utilise des extensions externes, raccordées par de la nappe souple, il sera bon d'utiliser du câble en nappe blindé.

Faute de pouvoir prendre ces précautions, il faudra éloigner fortement le ZX des récepteurs, ce qui, il faut bien l'avouer, n'est pas très pratique quand on veut en appliquer les possibilités au radioamateurisme.

Les perturbations principales se rencontrent surtout en ondes courtes. Sur VHF, les émetteurs-récepteurs en boîtier plastique (style IC2E, AR240, etc...) sont très perturbés. Les téléviseurs en bande I le sont aussi, ce qui est gênant pour la télévision d'amateurs.

Nous allons examiner d'autres problèmes rencontrés avec le système ZX et rechercher des solutions.

ECHAUFFEMENT

Le régulateur, alimenté sous 10 V en entrée sort 5 V sous 700 mA quand une RAM 16 K est utilisée. Il chauffe donc pas mal et son dissipateur est insuffisant. On peut le remplacer avantageusement par une plaque aluminium de 2 mm d'épaisseur et de dimensions respectables trouvant sa place sous le clavier d'origine.

Le circuit spécialisé, le micro-processeur et la ROM chauffent également un peu. Le tout, en enceinte close comme c'est le cas du ZX, fait que la température s'élève rapidement à l'intérieur de la machine. Le résultat ne se fait pas attendre et par moments, on obtient des pertes de programme, notamment quand le régulateur, trop chaud, se « coupe » par protection...

Cet échauffement a aussi une conséquence secondaire. Au bout de quelques minutes, il est fréquent de constater que la ligne supérieure de l'écran se met à trembler et finit par rester inclinée vers la droite. Ce défaut vient du circuit spécialisé (celui qui est le plus proche du modulateur) ; pour vous en convaincre, posez votre doigt en son centre. Non seulement vous ressentirez une légère brûlure, mais en plus vous verrez que l'affichage de la première ligne redevient correct.

La solution est simple. Il suffit de munir ce circuit d'un dissipateur qui pourra, là encore, être taillé dans un morceau d'aluminium de 2 mm. Sa dimension sera de 80x16 mm. On le fixera sans le coller (il y a risque de contraintes mécaniques) sur le dos du circuit ULA. Il tiendra de lui-même si on met de la graisse aux silicones sur toute sa surface et en l'appliquant fortement contre le circuit, mais il est préférable d'assurer sa fixation en y perçant deux petits trous traversés par des vis de 2 et fixées dans le circuit imprimé, aux environs de C1 TR1 vers le haut, et du marquage R30 vers le bas. La dissipation sera suffisante et tout entrera dans l'ordre.

On peut parfaire l'ensemble en créant une convection naturelle par perçage de petits trous sous le ZX et sur son capot supérieur, au moyen d'une mini-perceuse et d'un forêt de 1,5 ou 2 mm.

L'échauffement se manifeste aussi au niveau de l'alimentation du ZX. Le transformateur devient vite brûlant, surtout si on utilise l'imprimante. Rappelons que cette dernière doit être utilisée avec la «nouvelle» alimentation, censée sortir 1,2 A. En fait, ceci paraît bien présomptueux et les transformateurs semblent mal dimensionnés. Il vaut mieux prévoir une alimentation plus conséquente.

STOCKAGE SUR CASSETTE

Nous allons toucher là un point de polémique fréquente. La fiabilité des sauvegardes et chargements sur cassette. En principe, et quel que soit le micro-ordinateur, la cassette n'est pas le meilleur support pour contenir programme et données, mais si on la compare au prix des disquettes, au niveau amateur, et à moins de manipuler fréquemment des fichiers, la cassette est très économique.

En fait, les problèmes apparaissent surtout quand on passe des cassettes d'un magnétophone à un autre, si le réglage d'azimutage des têtes est différent ou si les vitesses de défilement ne sont pas identiques. Une période de signal à 3 000 Hz ne représente pas beaucoup d'espace sur une bande qui défile à 4,75 cm/s, et il vaut mieux la restituer correctement (les 0 et 1 sont représentés par 4 ou 8 périodes de 3 300 Hz, pour schématiser).

Choix de la cassette :

La cassette sera surtout bonne mécaniquement car il n'est pas nécessaire d'utiliser des cassettes au chrome. Point n'est besoin d'une large bande passante, les seuls signaux à enregistrer sont à 3 300 Hz. Il suffit d'une bonne dynamique et d'un bon défilement de la bande pour éviter les variations de positionnement de celle-ci face à la tête. On choisira des cassettes à flasques gaufrées et poulies de guidage.

On pourra avantageusement utiliser des cassettes 2 fois 5 ou 2 fois 10 minutes qui, parce qu'elles ne contiennent pas beaucoup de bande, seront entraînées correctement, même par des moteurs paresseux. Autre avantage : l'accès aux programmes sera beaucoup plus rapide, surtout si le magnétophone est sans compteur.

Pour les mêmes raisons, on évitera les cassettes C90 ...

Puisque nous en sommes aux cassettes, une précaution à prendre consiste à toujours doubler la sauvegarde d'un programme auquel on tient. Une cassette peut être accidentellement détruite ou perdue. Pensez aussi à casser les languettes de protection des cassettes qui peuvent toujours, si besoin est, être remplacées par des morceaux d'adhésif.

Prenez toujours grand soin de vos cassettes et évitez de les poser sur le téléviseur où sévissent des champs magnétiques non négligeables, ou bien sur le haut-parleur du magnétophone.

MAUVAIS CONTACTS

Que de problèmes, aussi, avec les contacts ! Ceux du clavier d'abord, pour les gens qui ont monté le ZX en kit, mais surtout ceux du connecteur d'extension. La mémoire extérieure s'y branche directement ou via le connecteur de l'imprimante. Dans les deux cas, la qualité des contacts est plus que douteuse et les programmes se perdent, au plus grand dam de l'opérateur un peu brusque avec le clavier

La solution passe par l'intégration de l'extension mémoire à un boîtier englobant le ZX, ou bien par une fixation de la mémoire sur le boîtier d'origine.

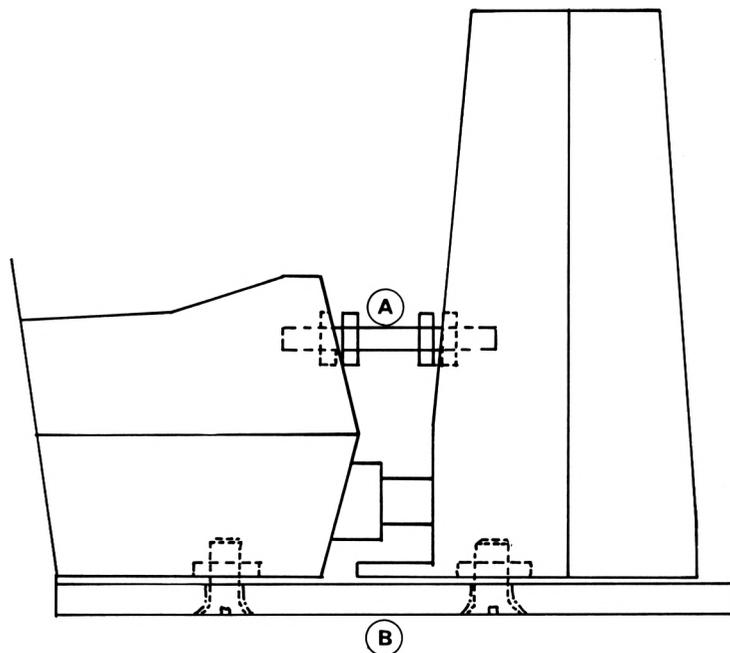
Cela peut se faire en perçant les boîtiers respectifs de la mémoire et du ZX de deux petits trous et en faisant passer des vis de 3,5 mm, assez longues, assurant une fixation efficace par un système écrou contre écrou (*Figure 3*).

Attention, enfin, aux contacts des jack (surtout celui qui assure l'alimentation) qui ne sont pas toujours de meilleure qualité.

Figure 3

Plusieurs possibilités :

- Système A : (écrou - contre écrou)
- Système B : fixation de l'ensemble ordinateur et mémoire sur un support en bois ou en plastique.



PERIPHERIQUES



*LE BOITIER ZX, MODIFIE AVEC CLAVIER MECANIQUE.
Noter le galvanomètre indicateur de niveau et l'inverseur «LOAD-SAVE».*

CHOIX DU MAGNETOPHONE

Il ne sera pas nécessairement de grande qualité BF, mais devra être choisi en fonction des facteurs suivants :

- la taille : le ZX n'est pas grand, donc rechercher un magnétophone qui ne soit pas trop volumineux.
- le compteur : la présence de cet accessoire sera très appréciée pour un repérage aisé des programmes sur la bande.
- alimentation : une alimentation secteur garantira des défaillances de piles (quoi de plus bête en effet, que de les voir faiblir en plein travail ...). La vitesse de défilement n'en sera que plus stable.
- sortie BF : il devra posséder une sortie haut-parleur supplémentaire pour être raccordé au ZX. En effet, les sorties «LINE» (bas niveau, 500 mV eff.) sont très insuffisantes pour pouvoir charger un programme (il faut au moins 3 V c/c sur l'entrée du ZX).
- entrée : une prise déconnectant le micro incorporé et prévue pour un micro extérieur sera nécessaire. Le ZX fournit 5 mV c/c sur sa sortie «SAVE». Si le magnétophone n'est pas équipé d'un compresseur de modulation, tant mieux !

Enfin, si le magnétophone possède un circuit de monitoring, il faudra éviter de laisser branchées simultanément les prises LOAD et SAVE du ZX, sans quoi il y aura des problèmes.

Précautions

Il faut entretenir régulièrement le magnétophone en nettoyant ses têtes et galet avec un coton tige imbibé d'alcool ou une cassette spéciale. Une démagnétisation périodique complètera l'entretien. Le bruit de fond et le niveau des aigues souffrent de l'encrassement et de la magnétisation des têtes.

Pour les duplications de programmes il faut éviter les copies de magnétophone à magnétophone, car il y a dégradation du rapport «signal-bruit». On passera, de préférence, par des opérations de LOAD/SAVE avec le ZX 81.

On pourrait améliorer la fiabilité de chargement en faisant passer les signaux à travers un filtre de bande centré autour de 3 300 Hz.

Un indicateur de niveau (à galvanomètre ou diodes électroluminescentes) pourra avantageusement être utilisé pour régler le volume de sortie du magnétophone quand on utilise une cassette enregistrée sur un autre appareil.

LE TELEVISEUR

Tous les modèles aux normes françaises conviennent. Chacun pourra choisir selon ses goûts ou disponibilités. Un écran d'une trentaine de centimètres semble être le meilleur compromis pour éviter une fatigue visuelle. Trop petit, il obligera à un effort, et trop grand, il faudra l'éloigner suffisamment. L'idéal est de posséder un téléviseur possédant une entrée vidéo (c'est le cas des moniteurs) ou une prise PERITEL. Dans ce cas, on obtient une image de bien meilleure qualité qu'en utilisant le modulateur UHF : meilleure définition, trame de fond moins prononcée, moins d'instabilités (le modulateur ayant tendance à dériver dans le temps).

RACCORDEMENT DU ZX A UNE ENTREE VIDEO

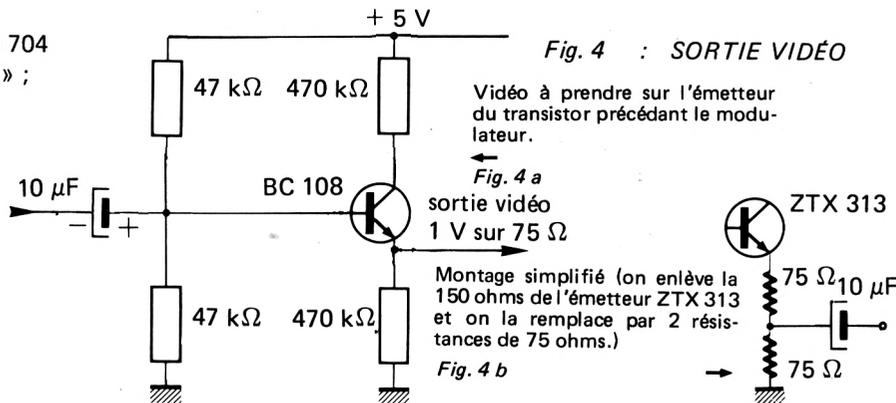
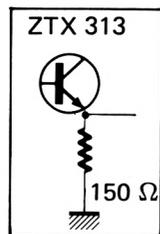
C'est vraiment très simple, surtout pour un TV à transistors qui demande moins de 1 V crête-crête de vidéo. Sinon on peut prévoir un ampli...

Voici le schéma de la modification (Figure 4).

Le transistor est celui qui précède le modulateur UHF.

Pour vous convaincre de la différence, il suffit d'afficher le petit «damier», (caractère graphique de la touche A) et de comparer la différence de résolution en passant de l'entrée antenne UHF à l'entrée vidéo.

```
10 FOR I = 1 TO 704  
20 PRINT «  » ;  
30 NEXT I
```



LES EXTENSIONS

Le système de base est très limité puisqu'il ne possède qu'une mémoire utilisateur inférieure à 1 K octets (compte tenu de la place occupée par les variables système) et pas de possibilités d'entrées-sorties (nous verrons que, malgré tout, il est possible en utilisant le langage machine, de disposer d'une ligne d'Entrée-Sortie par le biais de l'interface cassette).

Mémoire

L'extension à acquérir en premier lieu est la mémoire additionnelle. En effet, rares sont les programmes intéressants (mis à part ceux écrits en langage machine) qui tiennent sur quelques 800 octets disponibles.

On peut, bien sûr, se lancer dans la construction d'une extension mémoire, mais le jeu en vaut-il la chandelle, quand on connaît le prix des composants ? Par la production en grande quantité, le prix de vente d'une 16 K est inférieur ou très peu différent du coût de réalisation, par l'amateur, du montage équivalent.

Au cas où vous décideriez de réaliser une extension, voici quelques types de boîtiers mémoires utilisables, avec leur prix approximatif et leurs avantages ou inconvénients (*voir figure 5*).

TABLEAU COMPARATIF DE DIFFERENTS TYPES DE MEMOIRES					
	Format	Nb. pour 4 K	Prix unit.	Prix des 4 K	Observations
2114	1 K x 4	8	20	160	Statiques bon marchés Extension simple
444	1 K x 4	8	35	280	Statiques C.MOS faible consommation
4118	1 K x 8	4	60	240	Statique Extension simple
6116	2 K x 8	2	115	330	Statique C.MOS faible consommation, exten- sion très simple.

Ce tableau, établi sur le prix moyen d'une extension 4 K avec des composants différents, relevés au catalogue d'un même distributeur.

Cette petite extension était recherchée pour être implantée dans l'espace 8 K – 16 K, pour y loger des routines en langage machine.

L'extension la plus simple, en deux boîtiers, faible consommation, est aussi la plus chère... Celle réalisée au moyen de 4 boîtiers 4118 est un bon compromis simplicité/prix.

Notons qu'il n'est pas envisageable de construire ainsi une 16 K, surtout quand on tient compte du prix de vente dans le commerce, autorisé par des productions massives !

A titre d'exemple vous trouverez le schéma synoptique d'une extension de mémoire simple (voir figure 6).

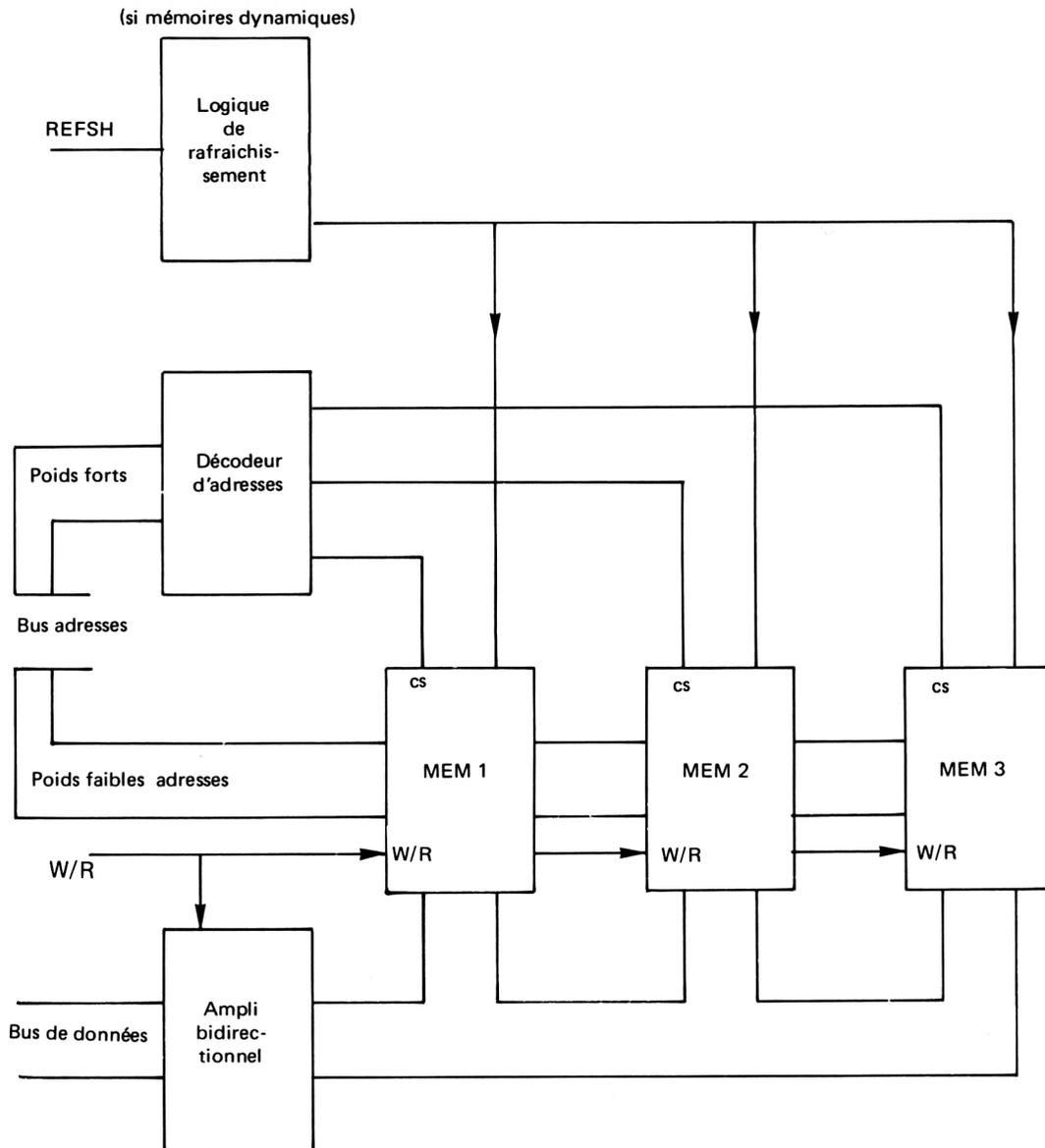


Fig. 6 : SYNOPTIQUE SIMPLIFIÉ D'UNE CARTE MÉMOIRE

Dans le choix de la taille de la mémoire, 16 K, 32 K, 48 K, 64 K, il ne faut pas perdre de vue certains points essentiels du système ZX et de son BASIC.

Il faut en effet savoir que le programme BASIC ne pourra jamais dépasser 16 K. Par contre, le volume de variables et données pourra être étendu. Nous savons que la mémoire globale est partagée entre le programme, le fichier d'affichage, les données et les zones de travail du système. Seules ces dernières pourront être « projetées » au-delà de 16 K. En résumé, si vous avez un programme de gestion avec des fichiers volumineux, la 48 K peut être intéressante. Si vous avez un long programme avec beaucoup de lignes et de commentaires et « Print » nombreux, sans pour autant avoir beaucoup de variables, votre problème ne sera pas résolu. Ce dernier cas est, malgré tout, assez exceptionnel.

Dans tous les cas, comme le système initialise la variable « mémoire disponible » (RAMTOP) au maximum à 16 K, il vous appartient de lui indiquer (par des POKE) que vous possédez une extension de taille supérieure. Notez enfin, que la 1 K interne ne vient pas s'ajouter à l'extension mémoire externe. Nous verrons comment il sera possible de la réutiliser.

Autres extensions

Si les applications de jeux vous intéressent, vous choisirez peut-être les cartes « graphiques » et « sonores ».

Les cartes graphiques existent en différentes versions :

— graphisme « haute résolution » permettant d'accéder à une meilleure définition du point élémentaire accessible à l'écran. Ainsi, de 64 x 44 de la fonction PLOT, on peut passer à 256 x 192. Ceci peut être intéressant pour des courbes mathématiques ou des dessins fins.

Graphisme par modification du générateur de caractères, qui permet de définir les caractères imposés à l'écran. Les caractères du ZX sont préprogrammés dans la ROM et ne peuvent être modifiés. Il existe des cartes, bâties tout simplement autour d'une RAM et des logiciels qui permettent de créer ses propres caractères. A titre d'exemple, voyez les résultats possibles (*figure 7*).

Quant aux cartes sonores, elles vous permettent de générer des sons complexes, de fréquence, hauteur et durée programmables, trouvant leur application dans les jeux, mais aussi, pourquoi pas, dans la génération de signaux CW ou RTTY.

Extensions importantes également : les interfaces entrée-sortie. Ce sont elles qui permettent le dialogue avec le monde extérieur. Il en existe de plusieurs sortes qui permettent de lire des signaux et d'en générer, signaux complexes produits par des convertisseurs numériques-analogiques, ou simples commandes de relais.

Elles sont bâties autour de simples bascules ou latches, ou autour d'outils spécialisés, tels que PIA, PIO, VART, ACIA, etc... Nous ne détaillerons pas le fonctionnement de ces outils, car là n'est pas notre but, mais nous dirons seulement que :

- PIA et PIO permettent les communications en parallèle,
- ACIA et UART autorisent les transformateurs série-parallèle avec conversion de vitesse ou de code (*voir application R. T. T. Y.*).

Avec ces outils on peut commander bien des travaux :

- programmation d'EPROM,
- mouvements d'antennes,
- passage automatique émission-réception,
- commande d'imprimantes.

Nous citerons enfin les convertisseurs numériques-analogiques ou analogiques-numériques qui permettent de générer ou d'acquérir des tensions et signaux divers. On peut ainsi développer des petits bancs de mesures pilotés par ZX 81.

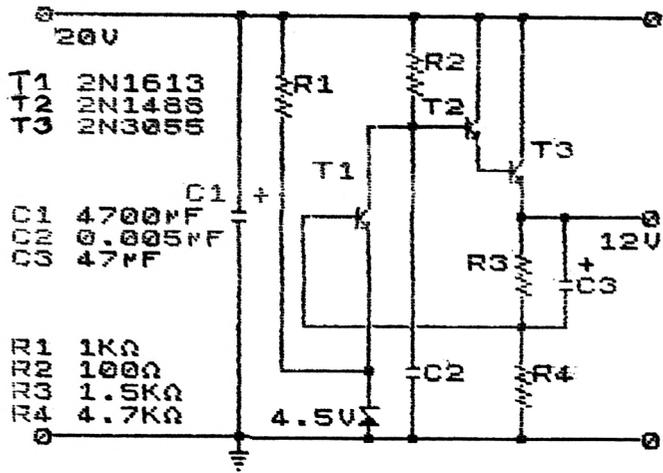


Fig. 7 : Schéma obtenu avec un générateur de caractères programmable (RAM implantée entre 8 K et 16 K)

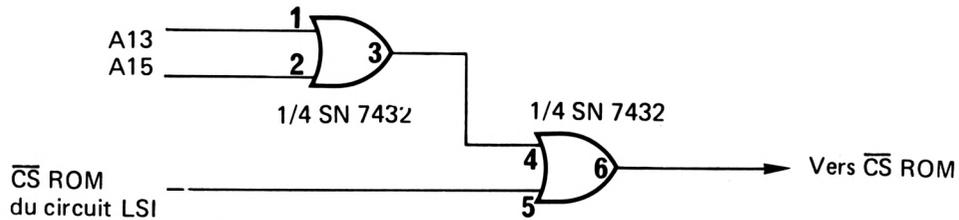


Figure 8

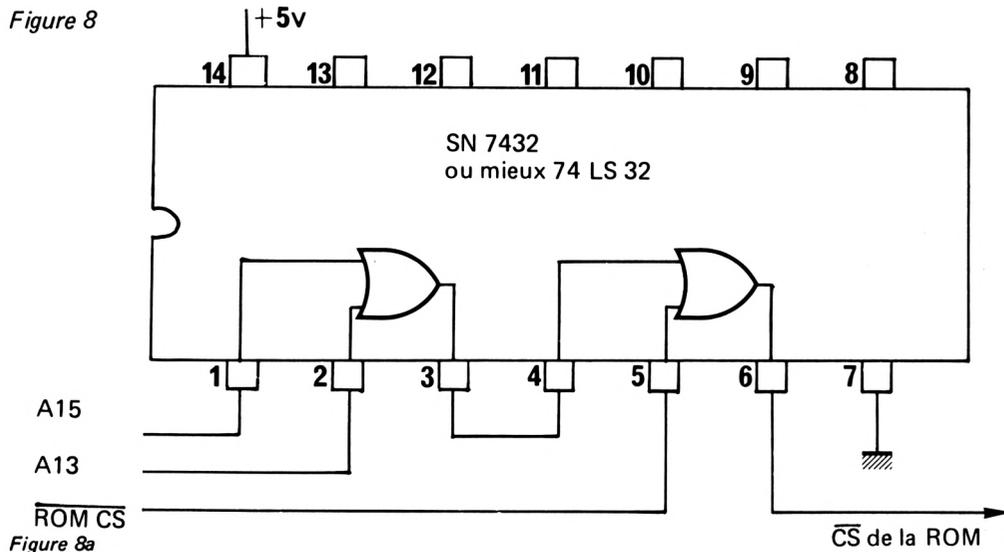


Figure 8a

Quand A 13 passe à 1 , on adresse la plage mémoire allant de 8 192 à 16 383.
 Quand A 15 passe à 1 , on adresse la plage mémoire allant de 32 768 à 65 535.
 Il faut donc supprimer la ROM à ces endroits.
 Par le \overline{CS} ROM venant du circuit LSI, on trouve la ROM à différents endroits:
 - de 0 à 8191: position normale de la ROM.
 - de 8192 à 16383: à supprimer.
 - de 32768 à 40959: à supprimer.

Donc la ROM doit être sélectionnée quand \overline{CS} ROM = 0 et A 13 = 0 et A 15 = 0.
 désélectionnée quand \overline{CS} ROM = 1 ou A 13 = 1 ou A 15 = 1.

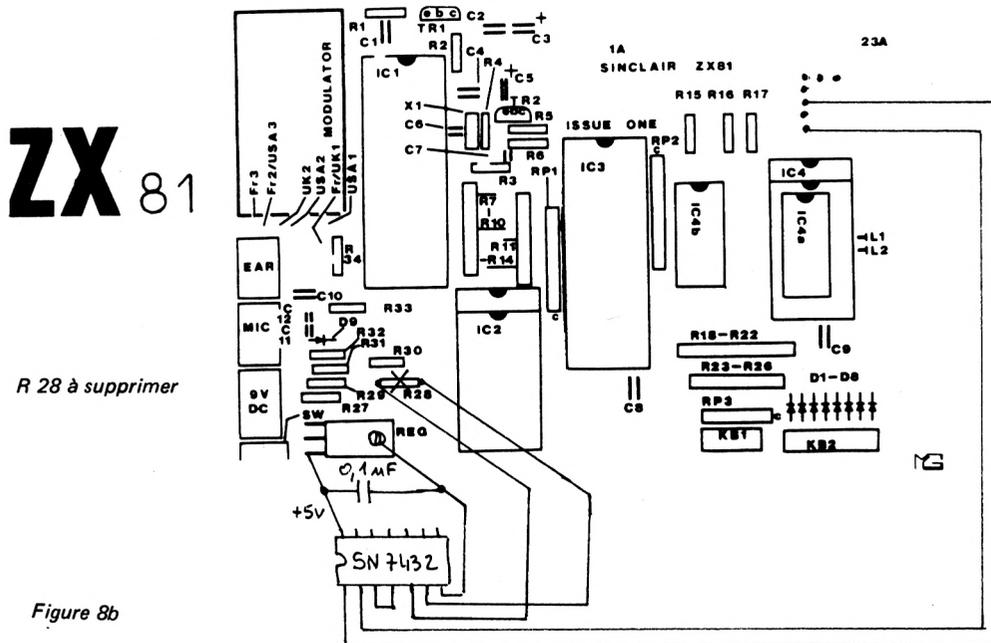


Figure 8b

On obtient des pages de 2 K octets:
 Y0 s'étend de 8192 à 10239
 Y1 s'étend de 10240 à 12287
 Y2 s'étend de 12288 à 14335
 Y3 s'étend de 14336 à 16383

A associer avec le schéma 74 LS 32

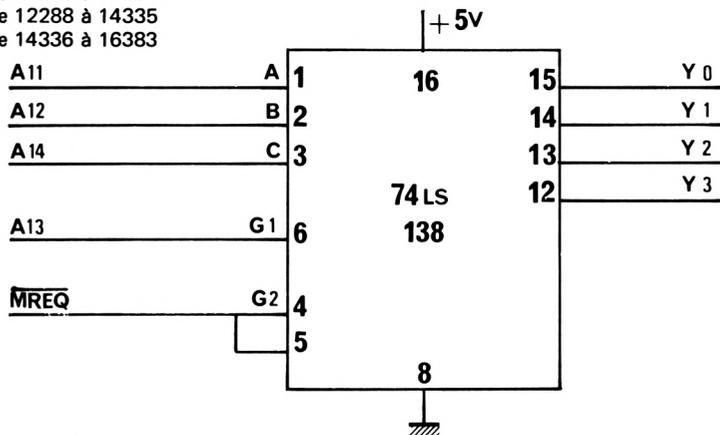


Fig. 9 : LE SYSTEME DE DÉCODAGE D'ADRESSES SIMPLIFIÉ

Imprimantes

Nous les évoquerons rapidement car elles complètent le système.

L'imprimante ZX «de recopie d'écran», allie faible coût et simplicité. Par contre, elle est bruyante et son papier, coûteux à l'emploi, pose des problèmes pour une utilisation plus professionnelle (petites factures, etc...). Pour ces raisons on peut être conduit à acquérir un autre type d'imprimante. Les moins chères, admettant du papier standard, coûtent environ 2 500 F. Il faut, dans ce cas, prévoir une interface assurant les compatibilités de code et standard. Le type «Centronics» est le plus répandu dans les imprimantes amateurs. Le type «RS 232» est aussi disponible et équipe plutôt des imprimantes d'une autre gamme.

Pour conclure notre chapitre «interfaces—extensions», il faut aborder le problème de leur implantation mémoire, le décodage d'adresses du ZX étant très incomplet. Chaque constructeur d'interface adopte un décodage qui lui est propre. Résultat : il faut se méfier des interfaces d'origines différentes qui peuvent se trouver dans la même zone mémoire et interférer...

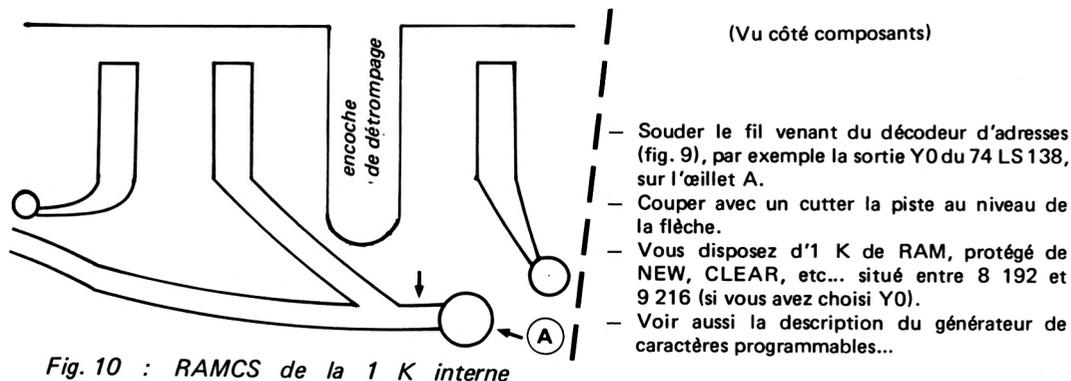
Nous vous donnons ici le moyen de modifier très simplement le décodage d'adresses du ZX, pour pouvoir le compléter de manière évolutive... En fait, le principe à retenir est que chaque carte interface aura son décodage d'adresses (*Figures 8, 8a et 8b, montage du 74LS32 dans le ZX ou 7432*).

Pour compléter ce décodage permettant de ne trouver la ROM qu'à un seul endroit (de 0 à 8191), on pourra avantageusement faire le montage de la *figure 9* qui permet de partager l'espace 8192 à 16383 en 4 zones de 2 K octets.

Le circuit intégré 74LS138 (tout comme le 74LS32 de la *figure 8*), trouve sa place facilement à l'intérieur du ZX. On le collera, pattes en l'air, avec de l'adhésif double face, sur le circuit imprimé du ZX. Les liaisons sont faciles à faire en utilisant du fil fin, tel que le fil à wrapper. Evidemment, ceci n'est valable que pour les petites extensions situées à l'intérieur du boîtier ZX (EPROM, Bip sonore, etc...).

Notons, à ce propos, que lorsque la RAM extérieure est utilisée, la RAM intérieure de 1 K octets ne sert plus à rien. On peut alors la récupérer et s'en servir pour y ranger des routines en langage machine par exemple quand on utilise le désassembleur décrit plus loin...

Il suffit de couper la piste RAMCS qui y arrive pour la brancher sur la sortie correspondante du décodeur d'adresses. L'endroit le plus facile pour pratiquer cette opération se situe au niveau du connecteur de sortie 2 A, face supérieure du circuit imprimé. Il suffit de couper la piste qui passe juste sous l'encoche de détrompage (*voir figure 10*).



LE BASIC DU ZX

Nous allons regarder de plus près quelques caractéristiques du BASIC du ZX.

Nous ne détaillerons pas les instructions, mais examinerons les plus particulières et nous vous renverrons, en guise d'exemple, à des extraits de programmes les utilisant.

Le BASIC est un BASIC «interprété». Disons, pour schématiser qu'un interpréteur lit à chaque exécution du programme les instructions pour les transformer en langage machine, par opposition à un compilateur qui agit en deux temps ; le premier étant de traduire, une fois pour toutes, en langage machine le programme qui est alors exécuté beaucoup plus rapidement.

Sans son langage, l'ordinateur n'est rien qu'un assemblage de composants ; c'est donc le langage qui lui confère la plupart de ses caractéristiques.

Le BASIC ZX occupe 8 K de la mémoire et il est logé en ROM entre 0 et 8191 (adresses décimales). Le générateur de caractères occupe les adresses de 7680 à 8191, soit 512 octets.

On voit donc que le langage lui-même occupe une place mémoire assez réduite, ce qui s'explique par l'utilisation de nombreux dérivements de programme, réutilisant au mieux bon nombre de routines.

Si, au point de vue vitesse, on ne peut faire beaucoup d'éloges, nous devons saluer au passage son analyseur syntaxique très pratique, interdisant l'introduction de lignes comportant une erreur. Une mention particulière également pour les facilités de gestion de curseur et d'édition. Quant au principe des mots-clés, codant chaque ordre BASIC, il est des plus pratiques lors de l'écriture d'un programme.

Quelques fonctions spéciales du basic Sinclair

La fonction DIM

Cette fonction initialise des tableaux à N dimensions.

Exemple :

DIM A (25) – tableau de 25 données

DIM A (10, 30) – tableau de 10 x 30 = 300 données

DIM C (10, 30, 2, 5, 6) – tableau de 18 000 données

DIM B\$ (10, 5) – tableau de 10 chaînes de 5 caractères.

On comprend facilement l'utilisation de tableaux à 1, 2 ou 3 dimensions, mais au-dessus, c'est un peu abstrait. Essayons donc par des exemples de rendre ceci plus abordable.

Un tableau à une dimension signifie que la variable est définie par un paramètre.

Exemple : DIM A (10) \Rightarrow A (1) = X A (2) = Y A (3) = Z etc...

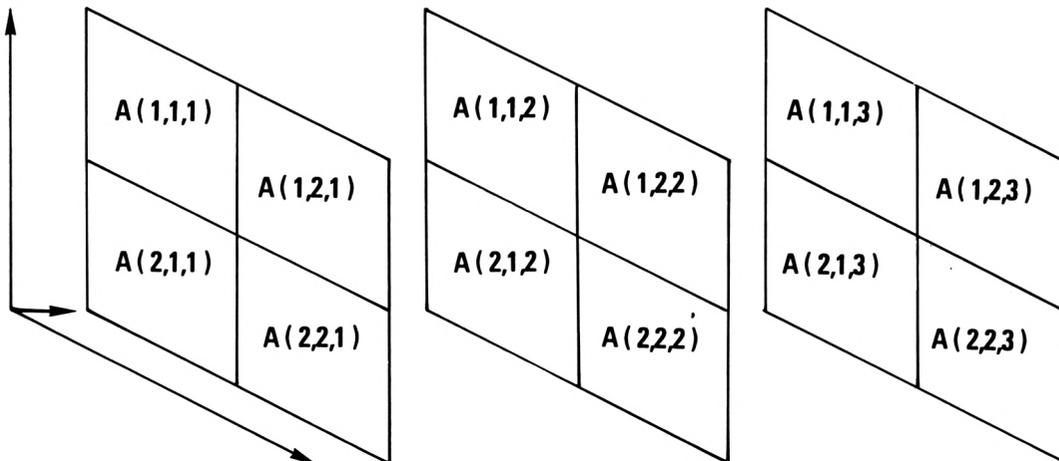
Pour un tableau à deux dimensions, une similitude peut être faite avec le plan où chaque point est défini par deux coordonnées.

Exemple : DIM B (2,2)

B(1,1)	B(1,2)
B(2,1)	B(2,2)

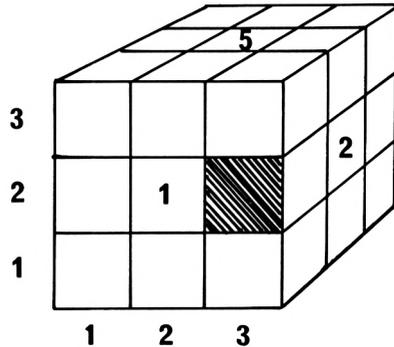
De même pour trois dimensions, le raisonnement peut se faire dans l'espace.

Exemple : DIM C (2, 2, 3)



Pour cette dimension, un exemple pourra peut-être mieux faciliter la compréhension. Prenons un objet bien connu des passionnés de jeux de patience : « Le Rubik's Cube ».

Pour entrer dans la mémoire de votre ZX la configuration de ce cube, plusieurs possibilités s'offrent à vous.



La première consiste à ouvrir un tableau à 1 dimension contenant $9 \times 6 = 54$ lignes. Chaque ligne représentant la couleur de chaque facette. Cependant, dans ce tableau il est difficile de s'y retrouver. En effet, il est nécessaire de numéroter chaque facette de 1 à 54.

Une autre possibilité est de dire que chaque petite facette peut être définie par ses coordonnées X et Y et par un numéro qui est celui de la face du cube considérée. Ceci nous donne donc 3 dimensions.

DIM A (3, 3, 6)
 X Y nombre de faces

La petite facette hachurée sur le dessin est définie par :

A (3,2,1)
 abscisse ordonnée numéro de la face du cube

Supposons maintenant que l'on désire mémoriser la configuration de plusieurs de ces cubes. Une solution serait d'ouvrir autant de tableaux à trois dimensions qu'il y a de cubes. Pourquoi ne pas créer plutôt une quatrième dimension ?

DIM A (3, 3, 6, 10)
 X Y face cube

La petite facette hachurée du troisième cube sera :

A (3, 2, 1, 3)

Qu'est-ce qui nous empêche alors de créer une 5ième dimension qui pourrait être alors par exemple le temps si les cubes sont manipulés et que l'on désire conserver une trace de la progression.

Exemple : DIM A (3, 3, 6, 10, 100)

nb de mouvements à faire.

Que devient la petite facette hachurée du 5ième cube après 32 mouvements ?

A (3, 2, 1, 5, 32)

Le découpage des chaînes de caractères

Le BASIC du ZX 81 permet un traitement particulier des chaînes de caractères. En effet, les fonctions telles que RIGHT\$, MID\$ ou LEFT\$ et bien d'autres n'existent pas mais sont remplacées par quelques astuces.

Faites sur votre machine :

LET A\$ = «SINCLAIR» puis N/L

La variable chaîne A\$ est initialisée.

Maintenant tapez : PRINT A\$ et «SINCLAIR» s'affiche sur l'écran.

PRINT A\$ (1 TO 8) a le même effet et signifie afficher A\$ du 1er au 8ième caractère.

PRINT A\$ (1 TO 5) donnera «SINCL» (affichage de A\$ du 1er au 5ième caractère).

PRINT A\$ (TO 5) est identique à l'exemple précédent.

PRINT A\$ (3 TO 8) aura le même effet que PRINT A\$ (3 TO) et donnera «NCLAIR».

Il découle de ceci que, lorsque la partie de la chaîne à considérer est à une extrémité, il est inutile de préciser dans l'expression entre parenthèses le numéro du premier ou du dernier caractère suivant le cas. Il n'est donc pas nécessaire de connaître la longueur de cette chaîne.

Exemple : PRINT A\$ (6 TO) ⇒ «AIR».

Ces particularités sont utilisées dans le programme «QTH AZI» qui est décrit plus loin.

Les opérateurs logiques

Ils sont très importants car ils permettent de simplifier l'écriture d'un programme. Ces opérations sont :

=, >, <=, >=, <, <>

On les utilise sous forme d'expressions entre parenthèses qui permettent d'effectuer des tests, des conditions. Ce qu'il faut retenir, c'est que, si cette expression qui se trouve entre parenthèses est vraie, le résultat est 1 et 0 dans le cas contraire.

Exemple : LET B = (A = 2)

B sera égal à 1 si A est égal à 2, sinon B = 0.

LET B = 10 * (A = 2)

B sera égal à 10 si A est égal à 2, sinon B = 0.

LET B = 23 * (INKEY\$ = «A»)

B sera égal à 23 si appui sur touche «A», sinon B = 0.

Il est également possible de compliquer l'expression logique, comme par exemple :

LET B = 10 * (INKEY\$ = «2» OR A > 3)

B sera égal à 10 si appui sur touche «2» ou si A > 3, ou les deux.

Les possibilités avec ces opérateurs logiques sont multiples et ne dépendent que de l'imagination de celui qui les utilise. Ils seront largement utilisés dans le programme «TV BV» décrit plus loin.

LES VARIABLES SYSTEME

Ce sont celles qui contrôlent le fonctionnement du système.

Elles sont, bien sûr, en mémoire vive, implantées de 16384 à 16508. Le livret du ZX nous enseigne succinctement le rôle de chacune d'elles ; nous ne considérerons ici que les plus intéressantes.

RAMTOP

C'est celle qui désigne le haut de la mémoire disponible. On conçoit donc qu'il est facile de modifier cette variable pour tricher sur la mémoire disponible. Son utilité est double :

- sauvegarder un court programme de taille inférieure à 1 K en version 1 K, tout en ayant la RAM 16 K «embrochée» sur le bus. L'intérêt est que le programme occupera une zone plus courte sur la bande et se chargera plus rapidement. Il suffit dans ce cas de faire un POKE en 16389 de valeur 69, suivi de NEW.

Le programme court qui avait été sauvegardé auparavant sans cet artifice, pourra être rechargé, puis sauvegardé à nouveau en version «1 K» cette fois.

A ce propos, si vous désirez connaître le nombre d'octets occupés par un programme, faites :

PRINT 53609 – USR 2583

- beaucoup plus utile est la modification de RAMTOP qui permet de ménager une zone protégée pour y mettre une routine en langage machine. Après cette modification de RAMTOP, toujours suivie de NEW, le programme ne sera affecté que par un RESET ou une coupure d'alimentation.

Il suffit d'imaginer que RAMTOP est un ascenseur qu'on peut envoyer à n'importe quel étage. Si on décide que, dans un immeuble de 6 étages, l'ascenseur ne montera que jusqu'au 4ième, les occupants des deux derniers étages ne seront jamais dérangés...

Il en va de même avec l'espace ménagé au-dessus de RAMTOP. On pourra même y ranger des données simples sous forme de variables qui pourront être transmises d'un programme, qui les écrira par POKE, à un autre qui les lira par PEEK.

D-FILE ET DF-CC

D-FILE est l'adresse de début du fichier d'affichage. On y trouve le code 118 qui marque le début du fichier. Nous savons que le système ZX ne possède pas de RAM vidéo, mais un espace (mobile), défini dans la RAM de travail, qui est utilisé pour contenir l'image envoyée à l'écran sous forme des codes «SINCLAIR» des caractères qui la composent : c'est le fichier d'affichage. Comme cette zone se déplace en fonction des modifications du programme (ligne qu'on ajoute ou qu'on retire), il faut que le système sache la retrouver. C'est le rôle du «pointeur» D-FILE. Nous pourrions donc en tirer parti et aller écrire par POKE, ou lire par PEEK dans cette zone. En version 16 K elle est composée de 24 x 33 caractères : 24 lignes de 32, plus un new-line. En prenant soin de ne pas «écraser» ces «new-line» (code 118), on peut donc écrire jusque sur les lignes 22 et 23, inaccessibles par le PRINT AT. Par PEEK on pourra même lire si un «point» du fichier d'affichage est allumé ou éteint.

Voyez l'exemple d'utilisation dans le programme TTY, pour écrire sur les 2 lignes du bas, le contenu du REM 90.

Vous pouvez, dans le petit programme suivant, qui noircit tout l'écran y compris les 2 lignes du bas (ce qui est mis en évidence en faisant apparaître un compte-rendu d'erreur), retrouver tout ce que nous venons de dire. Nous utilisons, ici DFCC qui est la position d'écriture dans le fichier d'affichage. La ligne 20 est essentielle. Sans elle on écrase les terminateurs de lignes («new-line») et on plante le système ; essayez de la supprimer pour voir ...

```
10 LET DFCC=PEEK 16398+256*PEE
K 16399
20 IF PEEK DFCC=118 THEN GOTO
35
30 POKE DFCC,126
35 LET DFCC=DFCC+1
40 GOTO 20
```

LAST-K

Mémorise, sur 2 octets, la «valeur» de la dernière touche enfoncée. Ceci est intéressant dans certains cas où on ne peut pas utiliser la fonction INKEY\$ à cause du temps de traitement.

```

10 IF INKEY#="" THEN GOTO 10
15 PRINT INKEY#,
20 PRINT PEEK 16421+256*PEEK 1
6420
25 GOTO 10

```

Le petit programme ci-dessus donne la valeur des touches pressées.

FRAMES

Est intéressante, car elle permet de «dater» un évènement (à condition d'être en mode «SLOW» et de ne pas utiliser de «PAUSE»). On peut s'en servir comme chronomètre pour déclencher un évènement après qu'un certain temps se soit écoulé. On travaille par bonds de 20 ms car FRAMES compte, en fait, les trames TV. Attention : le codage a lieu sur 2 octets, mais le bit de poids fort n'entre pas dans le calcul (il est utilisé par le système pour reconnaître mode SLOW ou mode FAST). Ne disposant dès lors que de 15 bits, il sera possible de compter jusqu'à 32767, ce qui représente 32767×0.02 s, soit environ 655 s ou 11 minutes. On initialise le chrono par POKE 16436, 255 et POKE 16437, 255, puis on lit ces deux adresses au moment voulu par PEEK.

Dans le cas d'un programme de poursuite de satellite, on pourrait ainsi, en consultant cette variable, envoyer un signal à intervalles réguliers, par exemple toutes les 2 minutes, pour faire bouger les aériens. En CW et RTTY il sera aussi possible de lancer des appels automatiquement toutes les N minutes ou secondes, en consultant FRAMES. Les applications ne sont qu'une question d'imagination.

Ci-dessous nous indiquons un programme qui permet de déduire le temps des instructions REM que vous pourrez ôter une par une, aux lignes 40, 50 et 60. Vous verrez qu'il vaut mieux éviter ainsi les REM dans les boucles répétitives...

```

10 POKE 16437,255
20 POKE 16436,255
30 FOR I=1 TO 1000
40 REM
50 REM
60 REM
100 NEXT I
110 PRINT "TEMPS = "; (65535-(PEEK 16436+256*PEEK 16437))*0.02;"
SECONDES"

```

S-POSN

Indiquent (en 2 octets) les numéro de colonne (16441) et ligne (16442). Vous pourrez en voir l'utilisation dans le langage machine du programme CW ou RTTY (décidant du CLS ou du Scrolling).

Voici, en BASIC, l'utilisation du numéro de ligne pour démarrer le Scrolling (essayez de modifier la ligne 20...).

```
10 FOR I=1 TO 50
20 IF PEEK 16442<3 THEN SCROLL
30 PRINT I
40 NEXT I
```

COORDS

Donnent les coordonnées en X et Y du dernier point allumé par PLOT. L'utilisation est surtout importante dans les jeux...

Pour en terminer avec les variables système, citons les deux octets libres 16507 et 16508, que l'on pourra utiliser comme «variables» auxiliaires. Leur avantage est qu'elles sont adressables en «indexé» quand on programme en langage machine, en partant de la valeur de base du registre IY qui est, rappelons-le, 16384.

Un peu à part, car ne faisant pas partie des variables système, on peut aussi voir une utilité aux 2 octets qui, pour chaque ligne, codent sa longueur. Cela permet de savoir, par exemple, quelle est la longueur d'un REM contenant du langage machine. On repère en mémoire les 2 octets codant la longueur (ils sont situés après les 2 octets codant le numéro de ligne) et on les lit par PRINT 256 * PEEK ADR + PEEK (ADR + 1), où ADR est l'adresse du premier octet codant la longueur.

Voilà, en quelques mots, nous avons fait le tour des variables système, les plus intéressantes. Il y a beaucoup à gagner à les utiliser.

INTRODUCTION AU LANGAGE MACHINE

Nous avons vu que le BASIC est un langage très pratique, utilisable aisément et qui est très facile à comprendre du fait de la grande similitude entre le terme anglais utilisé et l'action recherchée.

Cette grande facilité d'utilisation en fait un outil particulièrement adapté pour les programmes de calcul ou de gestion, où l'on tâchera d'exploiter au mieux toutes ses possibilités. Rappelons qu'il n'y a pas de bonne programmation sans une bonne analyse préalable du problème.

Nous savons qu'une des limites du BASIC ZX est sa lenteur d'exécution. Dans certains cas,

on peut être conduit à avoir besoin d'une grande vitesse. C'est un des avantages de la programmation en langage machine.

Si, de plus, on veut utiliser certaines ressources du ZX, comme ses 2 prises LOAD et SAVE, qui sont en fait 2 «ports» d'entrée-sortie vers le monde extérieur, ou si l'on désire implanter des périphériques nouveaux sur le système ZX (générateur sonore, carte graphique, circuits entrées-sorties), on est très rapidement conduit à utiliser les possibilités offertes par le langage machine.

Pour vous convaincre de la différence de vitesse d'exécution qui existe entre le BASIC et le langage machine, nous vous proposons d'introduire deux programmes. Le premier, en BASIC, est donné ci-dessous. Le second, en langage machine (son équivalent), sera décrit un peu plus loin. Tous deux vont remplir l'écran avec le caractère \$.

Seule différence, le temps d'exécution : 15 secondes pour le programme ci-dessous ; nous vous laissons découvrir le temps pour celui en langage machine décrit plus loin.

```
10 FOR I=1 TO 704
20 PRINT "$";
30 NEXT I
```

Auparavant, voyons comment il est possible, sur le ZX, d'accéder au langage machine.

Les ordres PEEK, POKE, USR

Pour lire un emplacement mémoire, on dispose sur le ZX de la fonction PEEK. Pour y écrire une valeur, nous utiliserons l'ordre POKE. Seule précaution, comme on travaille sur un octet, la valeur doit être comprise entre 0 et 255.

Ce dernier point nous conduit à évoquer le format d'écriture.

Que l'on travaille en BASIC ou en ASSEMBLEUR, le seul vrai langage compris par la machine est le binaire. Il y aura donc des étapes de transformations successives pour arriver au binaire (c'est le rôle de l'interpréteur BASIC ou du langage d'ASSEMBLAGE).

Pour déposer, par BASIC, une valeur binaire à un emplacement mémoire défini, nous devons utiliser la fonction POKE, suivie de la valeur en décimal. Si vous préférez utiliser l'hexadécimal, il vous faudra passer par un programme assurant la conversion hexadécimal – décimal.

Comme l'interpréteur BASIC ne peut pas deviner où se trouve le programme en langage machine, on le lui indique par son adresse de début par l'intermédiaire de la fonction USR. Il faut aussi savoir qu'on ne peut pas mettre du langage machine à n'importe quel emplacement.

Reconnaissance du langage machine

La fonction USR, suivie de l'adresse en décimal permet de prévenir l'interpréteur que ce qu'il va exécuter est en langage machine. La syntaxe sur le ZX est un peu particulière car on peut écrire :

```
GOTO USR n
LET L = USR n
RAND USR n
etc ...
```

où n (0 à 65535) est l'adresse de localisation de la routine en langage machine. On voit qu'on peut faire précéder la fonction USR par d'autres ordres BASIC. Si n est décimal, il sera arrondi à l'entier le plus proche, ce qui permet de définir n par une variable, si besoin est ...

Essayez de faire : RAND USR 3875 (n/L).

Vous venez de faire passer votre ZX en mode FAST, ce que vous pouvez vérifier en constatant que l'écran tréaute à chaque appui touche. L'effet de votre commande est identique à celui qu'aurait eu l'ordre BASIC «FAST» puisque 3875 (Ø F23 hexa.) est en fait l'adresse en ROM qui fait passer le ZX en mode rapide.

Faites maintenant : RAND USR Ø (n/L).

Vous avez ré-initialisé le ZX en appelant la routine ROM dans laquelle il entre à chaque mise sous tension.

Dernier essai : RAND USR 758.

Vous avez reconnu la fonction SAVE par les «dessins» qu'elle provoque sur l'écran.

Les routines que nous avons appelées se trouvaient toutes dans la ROM BASIC. Dans beaucoup de cas d'utilisation il n'en sera pas de même, et on fera appel au langage machine pour créer nos propres routines répondant à un besoin particulier.

Il faudra implanter ces routines à des endroits bien précis de la mémoire pour pouvoir, à coup sûr, retrouver leur adresse. Nous aurons recours à l'une des «astuces» suivantes :

- Implantation au-dessus de RAMTOP
- Dans une mémoire RAM annexe
- Dans une ligne REM
- Dans une chaîne de caractères.

Au-dessus de RAMTOP ou dans une RAM annexe

Cette solution offre la possibilité d'utiliser un programme langage machine sans avoir à le recharger en mémoire après l'introduction d'un ou plusieurs programmes BASIC différents. C'est intéressant car on voit que des programmes différents pourront laisser des résultats utilisables par d'autres programmes qui suivront et qui utilisent tous la même routine en langage machine.

Ceci est possible car tout ce qui est au-dessus de RAMTOP ou dans une RAM annexe située, par exemple, entre 8192 et 16383 n'est pas vu ou altéré par le BASIC, même si on fait RUN ou NEW.

Comment modifier RAMTOP ?

Le manuel du ZX est très clair à ce sujet. A la mise sous tension, selon la configuration mémoire, la valeur de RAMTOP est initialisée (pour 1 K on trouve 68 en 16389, et pour 16 K 128). Le ZX explore sa mémoire par une routine ROM, écrit à chaque emplacement une valeur donnée (la valeur 2) et relit ensuite chaque case. Dès qu'il ne trouve plus la valeur qu'il avait écrite précédemment, il décrète que le dernier emplacement mémoire est RAMTOP.

On trouvera ainsi 17408 pour 1 K et 32768 pour 16 K. Rappelons qu'on peut connaître RAMTOP en faisant :

PRINT PEEK 16388 + 256 * PEEK 16389

Pour modifier RAMTOP, on écrira les valeurs correspondantes dans ces deux adresses et on fera NEW.

Mais, lorsqu'on sauvegarde un programme, la K7 contient une «image» de la RAM vue par le système, ayant pour limite supérieure RAMTOP. Il est clair que, tout ce qui est au-dessus n'est pas sauvegardé sur K7, ce qui peut être gênant, d'où l'autre solution :

Dans une ligne REM ou une chaîne de caractères

Si on utilise une chaîne de caractères, il faudra que ce soit la première variable définie dans le programme, pour pouvoir la retrouver aisément dans la zone des variables, pointée par VARS. Ceci est un peu contraignant et nous préférons une ligne REM. Là encore, une contrainte, puisqu'il faut connaître à tout moment l'adresse du premier octet de la ligne REM.

REM est une instruction BASIC qui sert à introduire dans un programme des commentaires. A cet effet, le BASIC qui vient d'interpréter l'ordre REM sait qu'il ne doit pas chercher à comprendre ce qui suit cette instruction.

Ceci est intéressant car, puisqu'on peut mettre n'importe quoi derrière REM, pourquoi ne pas y mettre du langage machine ?

On peut connaître facilement l'adresse du premier octet de la ligne REM, surtout si elle est au début du programme. En effet, chaque ligne BASIC a le format suivant :

- numéro de ligne (codé sur 2 octets, octet le plus significatif en tête).
- longueur de la ligne, plus 1 octet (new-line), codée sur 2 octets (le plus significatif en tête).
- ordre BASIC, codé sur un octet (ex. : REM = 234).

Ces 5 octets représentant un format fixe, le sixième octet sera le premier qui nous intéresse. Sachant que le programme commence toujours en 16509, le premier octet d'une REM en tête de programme sera toujours en $16509 + 5 = 16514$.

Si, par la suite, on veut ajouter une autre REM contenant aussi du langage machine, il y a tout intérêt à le mettre à la suite car son adresse de début sera facile à déterminer, compte tenu de ce qui est expliqué ci-dessus.

Si la ligne REM est censée contenir le module en langage machine, il faut donc réserver suffisamment de place dans la REM. Ceci se fera tout simplement en tapant, derrière REM, autant de fois un caractère (quelconque), qu'il y a d'octets à réserver.

Pour faciliter ce travail, fort fastidieux pour les grands nombres d'octets, il vaut mieux passer en mode FAST. Pour mieux comptabiliser les octets, il est pratique d'utiliser des chiffres :

1 REM 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6

Noter qu'il vaut mieux prévoir plus d'octets que nécessaire, pour se garantir de modifications éventuelles.

Ceux qui possèdent une touche REPEAT sur leur machine en apprécieront grandement la présence. Pour tous les autres, nous proposons un programme qui va faciliter la tâche, puisqu'il crée automatiquement les REM à la dimension que vous choisissez, mais forcément en début du programme. Il se détruit automatiquement, ne laissant que la REM remplie de caractères ■

Ce programme pourrait être votre premier programme en langage machine. Respectez soigneusement son écriture. Ne changez absolument rien. Ligne 101 écrire «REM» sans mettre le M en vidéo inversée. Sauvegardez le sur une cassette par GOTO 100 (Voir dans le chapitre final le programme «REM»).

Quand un REM est ainsi créé, suivi de ses octets réservés, il faut alors le remplir.

C'est en décimal que vous introduirez vos octets réservés par POKE. Si vous désirez les introduire en hexadécimal, il faudra en assurer la conversion avant le POKE.

Voici des petits programmes qui pourront servir à introduire les différentes valeurs, pour 10 octets situés de 16514 à 16523.

```
10■ REM .....  
20 FOR I=16514 TO 16523  
30 SCROLL  
40 PRINT I,  
50 INPUT K  
60 POKE I,K  
70 PRINT K  
80 NEXT I
```

```
5■ REM .....  
10 FOR I=16514 TO 16523  
20 SCROLL  
30 PRINT I,  
40 INPUT A$  
50 PRINT A$  
60 LET L=(CODE A$(1)-28)*16+(C  
ODE A$(2)-28)  
70 POKE I,L  
80 NEXT I
```

Le premier programme fonctionne avec des valeurs introduites directement en décimal.

Le second programme assure la conversion hexadécimal vers décimal, avant de faire le POKE, pour les inconditionnels de l'hexadécimal !

Il existe bien sûr plusieurs manières différentes pour aborder ce problème et chacun verra en fonction de ses habitudes.

Evidemment, la meilleure solution consiste à utiliser un programme «ASSEMBLEUR» qui réalisera lui-même ce travail et vous permettra de rester au niveau des mnémoniques.

Une fois le REM rempli par le langage machine, il est conseillé de le sauvegarder sur cassette avant les premiers essais car, en cas d'erreur dans le langage machine, il y a risque de plantage et le BREAK sera inactif. Seule solution : débrancher l'alimentation ou faire RESET si vous avez installé un poussoir RESET (Figure 11).

Maintenant que nous savons où et comment implanter une routine en langage machine, nous allons achever la comparaison entre le programme BASIC et la version machine correspondante.

Pour imprimer un caractère, il existe une routine dans la ROM qui démarre avec l'instruction RST 0010. A ce moment, le code du caractère à imprimer est dans le registre A (l'accumulateur). On écrira donc le programme suivant :

```
En «assembleur»      LD A, 13      code du $
                     RST 10      impression
                     RET         retour au BASIC
```

On fera : 1 REM 1 2 3 4 (n/L)

```
POKE 16514,62 (n/L)
POKE 16515,13 (n/L)
POKE 16516,215 (n/L)
POKE 16517,201 (n/L)
```

```
Puis : 20 FOR I = 1 TO 704
        30 RAND USR 16514
        40 NEXT I
```

Avec ce procédé nous ne comparons que le temps d'exécution du «PRINT», puisqu'il subsiste le BASIC de la ligne FOR-NEXT. On pourra aussi tout écrire en langage machine.

Voici le programme correspondant. Exécutez-le par :

RAND USR 16514 (n/L)

après l'introduction de la dernière instruction de la liste. C'est bien plus long (20 s) car nous inter-prétons deux fonctions RAND USR au lieu d'une PRINT, malgré l'appel au langage machine.

```
10 REM .....14"00"16526
20 FOR I=16514 TO 16526
30 PRINT I,
40 INPUT K,
50 PRINT K,
60 POKE I,K
70 NEXT I
```

16514	17
16515	193
16516	2
16517	27
16518	122
16519	179
16520	200
16521	62
16522	13
16523	215
16524	24
16525	247
16526	201

La machine affiche les adresses (colonne de gauche), vous introduisez les données (colonne de droite). Après la dernière, vous faites directement RAND USR 16514 (n/L).

Le temps ? Moins d'une seconde pour remplir l'écran ...

Comparez les temps d'exécution des trois programmes. Vous êtes convaincus ? On continue !

Avant d'aller plus loin dans l'écriture de nos programmes, il sera nécessaire de voir un peu quel est le jeu d'instructions du microprocesseur dont nous disposons : le Z 80.

Nous ne détaillerons pas tout le jeu d'instructions, ni leur mode d'utilisation : ce n'est pas le but de ce livre, de même que nous n'entrerons pas dans l'architecture détaillée du Z 80 et nous nous contenterons d'en donner les grandes lignes.

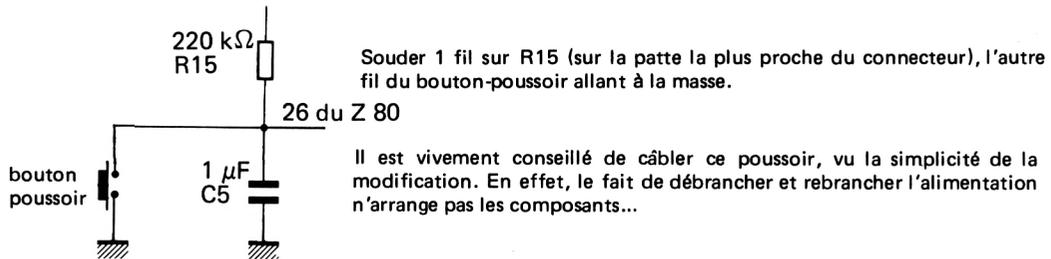


Fig. 11 : CABLAGE D'UN POUSSOIR RESET

LE MICROPROCESSEUR Z 80

C'est un peu le cerveau de notre ZX (Figures 12 et 13).

Le Z 80 est un produit dérivé du 8080 dont il a conservé une compatibilité importante en ce qui concerne les instructions. C'est un microprocesseur de la famille «8 bits» (tout y est organisé en «mots» de 8 bits ou octets, byte en anglais).

Il possède plusieurs registres qui sont :

A l'accumulateur et F le registre d'états
B et C
D et E
H et L

Ces registres peuvent être appariés en BC, DE, HL (et aussi dans une certaine mesure, AF). On peut alors les utiliser sur 16 bits. Ils sont tous doublés par des registres secondaires également appairables (B'C' D'E' A'F').

Une instruction permet l'échange des contenus respectifs entre registres «primaires» et «secondaires».

Viennent ensuite des registres d'index IX et IY (sur 16 bits), un registre pointeur de pile SP (stack pointer, 16 bits), un compteur PC (program counter, 16 bits).

Il existe aussi un registre I (vecteurs d'interruptions) et R (rafraichissement mémoire).

En décrivant ainsi le Z 80, nous passons sous silence l'unité arithmétique et logique, le registre instructions, l'unité de contrôle..., mais en fait, nous nous plaçons côté utilisateur.

Il faut enfin savoir que le bus de données est composé de 8 lignes D0 à D7 et le bus adresses de 16 lignes repérées, A0 à A15, et qu'il existe aussi des signaux de contrôle très importants, tels que $\overline{\text{HALT}}$, $\overline{\text{NMI}}$, $\overline{\text{IORQ}}$, etc... dont nous n'aurons pas à nous soucier sur le plan programmation et que nous laisserons volontiers «tomber» (remarquons simplement au passage qu'ils sont actifs au niveau bas).

Nous détaillerons, par contre, un peu plus un registre bien particulier, qui est le registre F (flag register) ou registre d'états. Il contient des «indicateurs» (on dit aussi des «drapeaux» ou «flags») que nous utiliserons souvent dans nos programmes.

Pour fonctionner, ce microprocesseur doit être associé à une mémoire morte (ROM), une mémoire vive (RAM) et un circuit d'horloge. Sur le ZX la fréquence de l'horloge est 3,25 MHz, car il est équipé d'un Z 80 A fonctionnant jusqu'à 4 MHz.

Rôle des Registres

Le registre double qui possède le plus d'instructions est HL. C'est lui qu'on utilisera de préférence pour manipuler des adresses. Les registres BC et DE peuvent aussi être utilisés à cet effet, mais sont moins pratiques.

Nous ferons, bien sûr, très souvent appel à l'accumulateur (A) sur lequel se font bien des opérations.

Sur le ZX, nous rencontrerons quelques contraintes lors de la programmation :

IX et IY doivent être manipulés avec précautions. IY contient 16384 (4000 H), qui est la base des variables système. IX est utilisé pour l'affichage (son contenu est différent en mode lent ou en mode rapide).

AF est également utilisé pour l'affichage.

On ne pourra donc pas vraiment faire «n'importe quoi» avec ces registres et il faudra bien garder cela présent à l'esprit. Notamment, penser à remettre 16384 dans IY si on altère sa valeur....

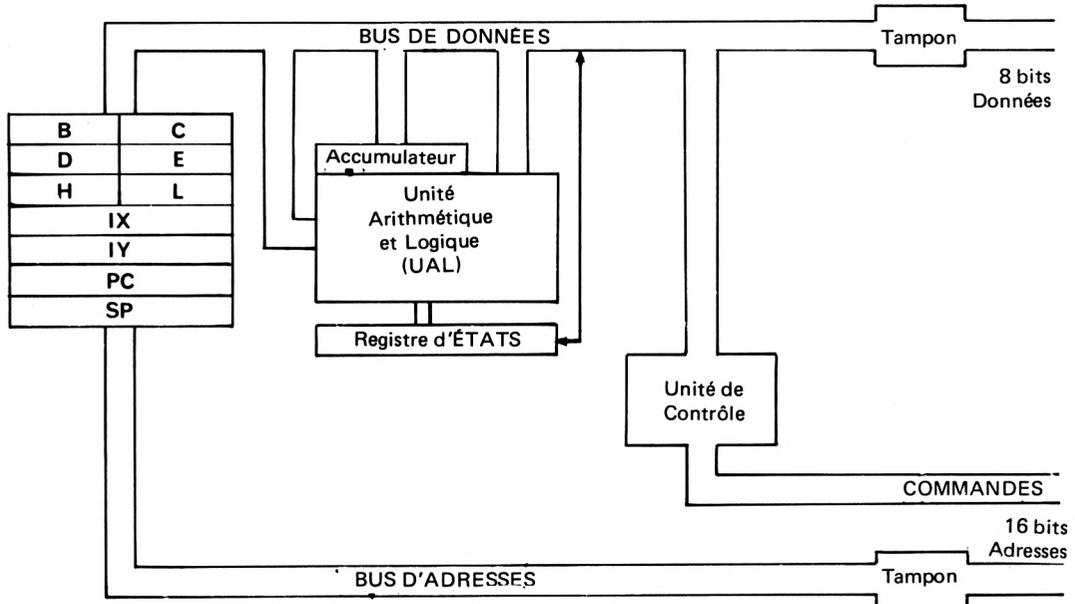


Fig. 12 : ARCHITECTURE ULTRA-SIMPLIFIÉE DU MICROPROCESSEUR Z 80

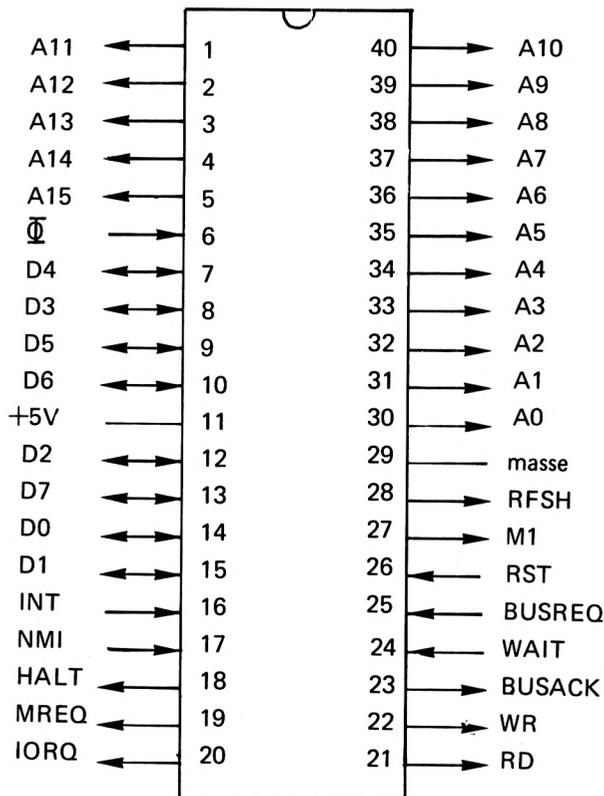


Fig. 13 : BROCHAGE DU Z80

Les registres I et R sont également utilisés par le système ZX.

I contient l'adresse haute du générateur de caractères ($1E_H$).

R est utilisé pour compter les caractères présents sur une ligne envoyée vers le téléviseur et, après 32 caractères, il y aura génération d'une interruption.

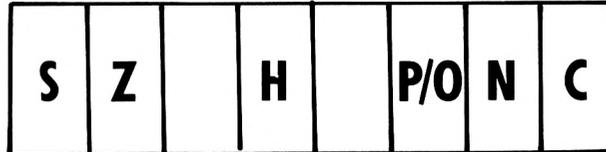
Les interruptions sont donc pratiquement inutilisables (NMI, HALT sont utilisées pour l'affichage).

On voit donc que, dans certains cas, l'architecture du ZX et la conception de sa ROM pourront nous limiter pour la programmation en langage machine, ce qui est d'ailleurs précisé dans le manuel de base.

Le Registre d'Etats

C'est un registre bien particulier, dont les bits sont indépendants et possèdent chacun un rôle «indicateur» bien précis.

Pour l'utilisation, seuls 6 sur les 8 sont importants et, nous verrons dans les programmes machines, présentés dans cet ouvrage, que 2 surtout sont très utilisés «Z et C».



S	sign	indicateur de signe
Z	zero	indicateur de zéro
H	half-carry	demi-reteneue
P/O	parity/overflow	parité-débordement
N		addition/soustraction
C	carry	reteneue

Nous disons qu'un bit est «positionné» lorsqu'il est à 1. Sans entrer dans les détails, pour lesquels nous invitons nos lecteurs à nous reporter à un ouvrage spécialisé sur la programmation du Z 80, voici les caractéristiques essentielles des indicateurs :

S bit de signe

Dans la notation en complément à deux, le bit 7 est le bit de signe. S'il est à 0, le nombre représenté est positif, s'il est à 1 il est négatif.

Z bit de zéro

Quand on fait une opération ou un mouvement de données, l'octet qui en est l'objet peut être nul. Dans ce cas, le bit de zéro passe à 1. Il est aussi positionné, lors des tests de comparaison si on a égalité.

Il faut faire très attention car toutes les opérations n'affectent pas le bit de zéro et une des erreurs les plus courantes consistera à le tester alors que l'opération précédente était sans effet sur lui ...

H demi-reteneue

Si l'on coupe un octet en 2, on obtient 2 groupes de 4 bits. Lors de certaines opérations (notamment addition et soustraction), il peut y avoir report du bit 3 «poids fort» du quartet «bas» vers le bit 4, «poids faible» du quartet «haut». Dans ce cas le bit H est positionné.

P/O parité-débordement

La partie d'un octet est déterminée en comptant le nombre de 1 qu'il contient. Si ce nombre est pair, on dit que la «parité est paire» et on positionne le bit «P/O». Les tests de parité sont surtout utilisés lors des transferts de caractères.

Quant au débordement, il est signalé quand le résultat d'une opération vient modifier, par erreur, le bit de signe (bit 7).

N soustraction

N'est pas directement utilisable par le programmeur. Mis à 1 après une soustraction il est surtout vu par le microprocesseur.

C retenue

Cet indicateur est très important et il sera souvent utilisé. Lors d'une opération, il indique la retenue. Il est affecté par toutes les opérations arithmétiques, mais aussi par certaines opérations logiques. On tirera parti de ce dernier point pour l'effacer implicitement.

LE JEU D'INSTRUCTIONS

Si le ZX possède un jeu d'instructions relativement réduit (environ 50), il n'en est pas de même avec le Z 80 qui possède un jeu d'instructions très puissant et très étendu.

Nous ne les étudierons pas une par une car il existe de très bons ouvrages en la matière, et nous nous contenterons de retenir les principales, ou celles qui sont le plus souvent utilisées.

Les instructions Z 80 peuvent requérir 1 à 4 octets. Heureusement, beaucoup d'instructions puissantes sont, nous le verrons, codées sur 1 octet, ce qui garantit une grande simplicité de programmation et une rapidité d'exécution. En programmant, il faudra essayer de conserver cela présent à l'esprit pour utiliser à bon escient ces instructions.

Nous allons introduire ici une notion particulière : le mnémotique.

C'est une appellation qui permet de retenir plus facilement une instruction que ne le permettrait son code hexadécimal ou décimal. Les mnémotiques sont dérivés des noms anglais des instructions.

JP pour jump (saut), LD pour load (charger), RET pour return (retour d'un sous-programme). Ainsi, RET correspond à C9 (hexa.) ou 201 (décimal).

JRNZ saut relatif si non-zéro correspond à 20, d (hexa.) et 32, d (décimal).

Ce sont ces mnémotiques qui sont utilisés lors de la programmation en ASSEMBLEUR. Lorsque nous programmerons en langage machine, nous écrirons ces mnémotiques sur le papier et nous porterons, en face les codes correspondants, en hexadécimal ou en décimal. Il est aussi indispensable de réserver de la place pour mettre des commentaires, permettant de s'y retrouver plus facilement, en reprenant le programme quelques mois plus tard. Selon ses habitudes, chacun pourra adapter le document ci-dessous, de même qu'il pourra adopter une écriture en décimal ou en hexadécimal.

Adresses	Mnémotiques	Code Machine	Commentaires
16514	LD A, (16442)	58, 58, 64	charge dans l'accu le numéro de ligne et effectue un retour si égal à 4. sauvegarde contexte sinon
16517	CP 4	254, 4	
16519	RET NC	208	
16520	PUSH DE	213	

Ce sont ces mêmes mnémoniques que l'on retrouve dans les dernières pages du manuel de programmation du ZX.

Nous allons essayer de répartir en «grandes familles» les principales instructions Z 80.

- instructions arithmétiques
- instructions logiques
- positionnements et tests de bits
- transferts de données
- déroulements de programme
- opérations d'entrées–sorties
- comparaisons
- diverses ...

Instructions arithmétiques :
ADD, ADC, SBC, SUB, INC, DEC.

Instructions logiques
AND, OR, XOR.
Rotations RL, RR
Décalages SLA, SRA, SRL

Positionnements et tests de bits
SET, RES, BIT.

Transferts de données
LD, LDD, LDDR, LDI, LDIR.

Déroulements
JP, JR, CALL
RST
RET

Entrées–Sorties
IN, OUT
IND, INDR, INI, INIR
OTDR, OTIR, OUTD, OUTI

Comparaisons
CP, CPD, CPDR, CPI, CPIR

Diverses
NOP, NEG, SCF, CCF, CPL, DAA
PUSH, POP

Commentaires sur quelques instructions les plus utilisées

Instructions arithmétiques

Instruction ADD

Cette instruction réalise une addition entre l'accumulateur (A) et un opérande, le résultat étant remis dans A. Réalise également l'addition de la paire de registres HL avec une autre paire, le résultat se trouve dans HL.

Exemple :

MNEMONIQUE	CODE DECIMAL
ADD A, nn (donnée 8 bits)	198 nn $A + nn \Rightarrow A$
ADD A, A	135 $A + A \Rightarrow A$
ADD A, H	132 $A + H \Rightarrow A$
ADD A, L	133 $A + L \Rightarrow A$
ADD HL, HL	41 $HL + HL \Rightarrow HL$
ADD HL, DE	25 $HL + DE \Rightarrow HL$

La même instruction existe pour les registres doubles IX et IY.

Exemple :

ADD IX, DE	résultat dans IX
ADD IY, BC	résultat dans IY

Les instructions sur IX et IY se codent sur deux octets, le premier est 221 pour les instructions sur IX et 253 sur IY.

Instruction ADC

Réalise la même fonction que ADD avec en plus la valeur du bit retenue («carry») qui est ajoutée au résultat.

Instruction SUB

Soustrait à A la valeur de l'opérande, le résultat est dans A. L'opérande peut être, tout comme l'instruction ADD, une donnée, un registre ou le contenu d'une mémoire dont l'adresse est donnée par HL ou par IX ou IY.

Instruction SBC

Soustrait à A la valeur de l'opérande et la retenue, le résultat est toujours dans A. Même opération avec les registres doubles HL, BC, DE et SP avec résultat dans HL.

Exemple :

SBC A,B	\Rightarrow	$A - B - \text{Carry} \Rightarrow A$
SBC HL, DE	\Rightarrow	$HL - DE - \text{Carry} \Rightarrow HL$

Instruction INC

C'est une incrémentation. On ajoute 1 à un registre, une mémoire ou une paire de registres.

Instruction DEC

Décrémentation d'un ou d'une paire de registres ou bien d'une mémoire.

Instructions logiques

Instruction AND

Cette instruction est le «ET» logique entre chaque bit de l'accumulateur et un opérande qui peut être :

- une donnée
- un registre
- le contenu d'une mémoire

Un exemple permettra de clarifier ceci :

Supposons que dans l'accumulateur nous ayons AA_H.

AA (en hexadécimal) = 170 (en décimal) = 1 0 1 0 1 0 1 0 (binaire).

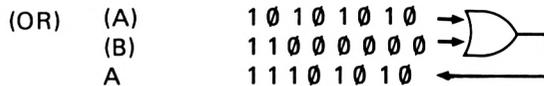
Et dans B, C0_H = 192_D = 1 1 0 0 0 0 0 0_B.

Que donnera AND B : (le résultat est dans A) ?



Instruction OR

OU Logique



Instruction XOR

OU exclusif

La configuration 1 XOR 1 = 0 contrairement au OU



Instruction de rotations

Il s'agit de décalages de bits de registres ou mémoires vers la droite ou vers la gauche à travers ou non le bit de retenue du registre d'état.

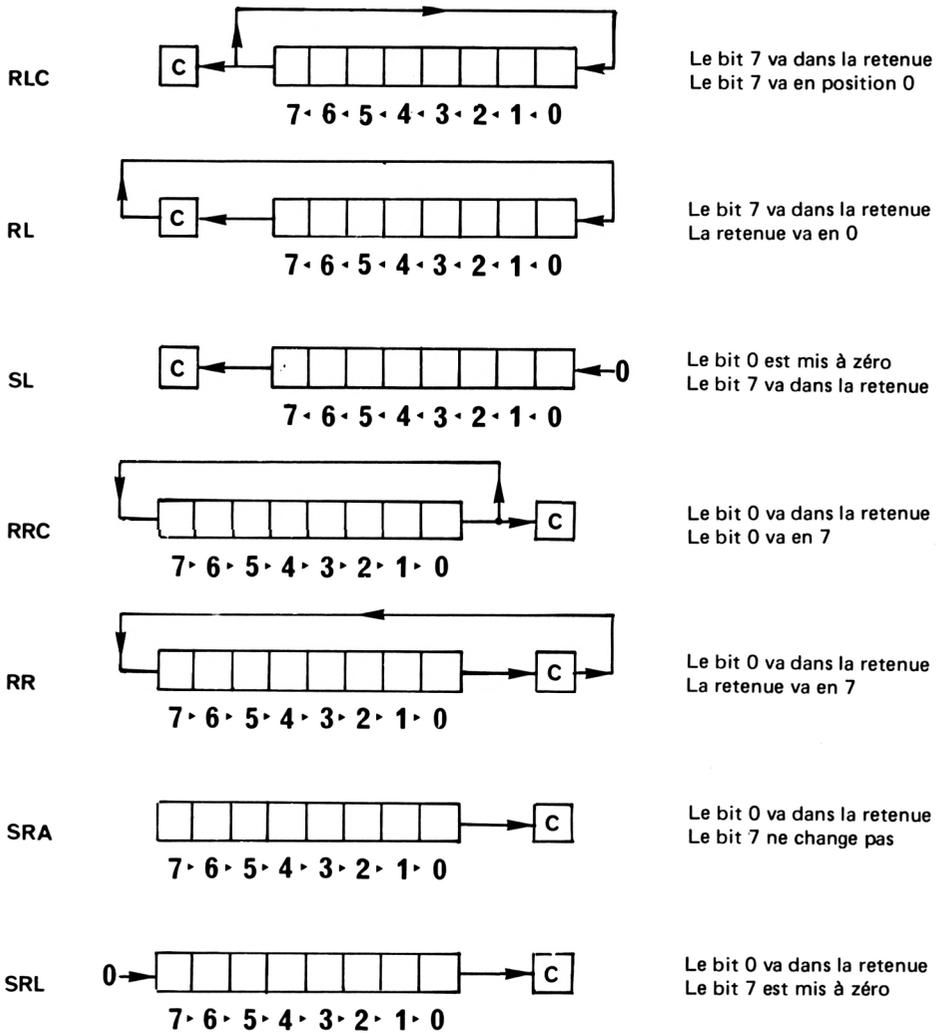
Positionnement et Test de bits

SET : permet de positionner à 1 un bit parmi huit d'un registre ou d'une mémoire.

RES : remet à zéro un bit.

BIT : teste le bit désigné d'un registre ou d'une mémoire. Le résultat positionne le bit «Z» du registre d'état. Si le bit testé est à zéro, alors le «Flag Z» est à 1 et inversement.

ROTATIONS



Transfert et Données

Plusieurs possibilités sont disponibles avec l'instruction «LD» qui vient de «LOAD». Il serait fastidieux de toutes les détailler. Ce qu'il faut retenir, c'est que lorsque vous écrivez par exemple :
LD X, Y

X est le destinataire et Y l'origine.

Exemple : LD B, H

signifie que le contenu de H est transféré dans B. Il existe aussi quelques instructions de transfert puissantes qu'il est bon de connaître.

LDD : permet de transférer un octet de l'adresse mémoire pointée par HL vers l'adresse mémoire pointée par DE. A la fin de ce transfert, HL, DE et BC sont décrémentés et quand BC = 0, le bit «P/O» du registre d'état sera remis à zéro.

LDI : même fonction que LDD mais les registres HL et DE sont incrémentés.

LDDR : sert à transférer des blocs de données. L'octet pointé par HL est transféré vers la mémoire pointée par DE. HL, DE et BC sont décrémentés et le cycle reprend jusqu'au moment où BC = 0. Ceci permet donc de transférer au maximum 65536 octets.

LDIR : même fonction que LDDR mais HL et DE sont incrémentés.

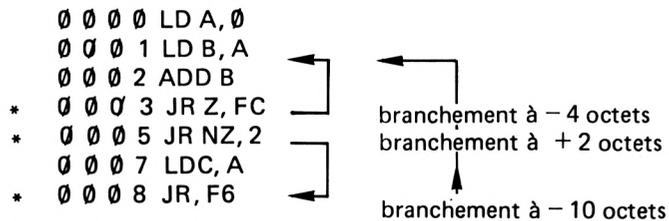
Instruction de saut

Qu'ils soient conditionnels ou inconditionnels, le Z 80 dispose de 2 types de sauts :

- les sauts absolus
- les sauts relatifs

Dans les instructions de saut absolu, par exemple JP nn, nn est l'adresse où doit s'effectuer le saut. (Ex : JP Z, OB84). Les instructions de saut relatif du type «JR, n» donnent pour n le nombre d'octets en amont ou en aval où doit se poursuivre le programme. Le déplacement maximum avec ce type d'instructions est de 128 octets, le bit 7 donnant le signe (+ ou -).

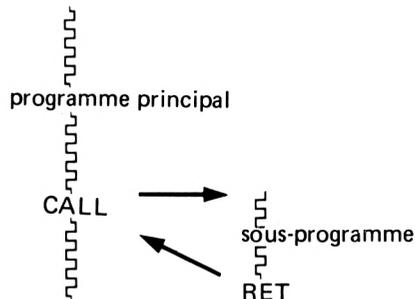
Exemple de programme :



L'instruction Call

Elle peut être inconditionnelle ou conditionnelle. C'est l'appel à un sous-programme. L'adresse de retour est sauvegardée dans la pile afin de revenir au programme principal dès que l'instruction RET est trouvée dans le sous-programme.

Exemple :



Dans chaque sous-programme, il peut y avoir aussi des CALL à d'autres sous-programmes, mais il doit y avoir autant de RET que de CALL.

L'instruction RST

Même utilisation que le CALL, seulement l'adresse de branchement est prédéterminée. Il y a 8 RST possibles.

RST 0000, 0008, 0010, 0018, 0020, 0028, 0028, 0030, 0038.

Une de ces instructions la plus utilisée dans les programmes de ce livre est RST 10 qui est la routine d'affichage de caractères à l'écran.

Instructions d'entrées—sorties

Instructions IN et OUT

IN permet de collecter des informations d'un périphérique. OUT permet d'envoyer des informations vers un périphérique.

Liste des instructions IN et OUT			
Mnémonique	Registre d'entrée (IN) ou de sortie (OUT)	Adresse haute du port	Adresse basse du port
IN A, nn	A	A	nn
IN A, (C)	A	B	C
IN H, (C)	H	B	C
IN L, (C)	L	B	C
IN B, (C)	B	B	C
IN C, (C)	C	B	C
IN D, (C)	D	B	C
IN E, (C)	E	B	C
Idem pour les instructions OUT			

Il existe également des entrées ou des sorties par blocs.

INI : permettant l'entrée de données avec incrémentation d'un pointeur de rangement.

INIR : même utilisation que précédemment, seulement son fonctionnement est automatique jusqu'à remplissage de la zone de rangement.

IND : Idem de INI mais avec décrémentation.

INDR : Décrémentation et automatisme.

Il y a également le même type d'instruction pour les sorties (OUT).

Instructions de comparaison

Elles permettent des comparaisons entre le contenu de l'accumulateur et une donnée, un registre ou une mémoire.

Exemple : CPH, CP nn, CP (HL)

Cette comparaison est en fait une soustraction entre A et la donnée. Le résultat n'est pas conservé mais les «flags» sont affectés. On peut aussi savoir si la donnée est inférieure, supérieure ou égale au contenu de l'accumulateur.

Exemple :

Savoir si A = 10	CP10
	JRZ «égal à 10»
Savoir si A ≥ 10	CP 10
	JRNC «supérieur à 10»
Savoir si A < 10	CP 10
	JRC «inférieur à 10»

Tout comme les instructions d'entrée, sortie, les comparaisons peuvent se faire par blocs : CPD, CPI, CPIR, CPDR

Autres instructions

- NOP : non opération : ne sert qu'à faire de petites temporisations.
1 NOP dure 1,23 microsecondes (sur le ZX).
- NEG : Complément à 2 de l'accumulateur (les «flags» sont affectés).
Ex : A = 00110000 ⇒ NEG : 11001111
- SCF : met à 1 la retenue (carry)
- CCF : met à 0 la retenue
- CPL : complément de l'accumulateur
- DAA : ajustement décimal de l'accumulateur (BCD)

Exemple : en binaire codé décimal (BCD) :

0 ⇒ 0 0 0 0 0 0 0 0
41 ⇒ 0 1 0 0 0 0 0 1
88 ⇒ 1 0 0 0 1 0 0 0

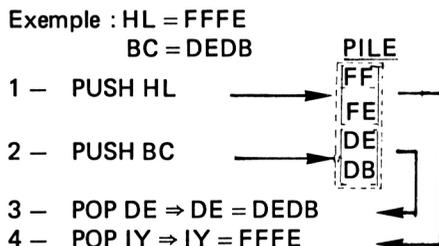
Donc, si dans l'accumulateur on a :

A = 0 0 0 0 1 1 0 0 qui représente 12(D) ou C(H).

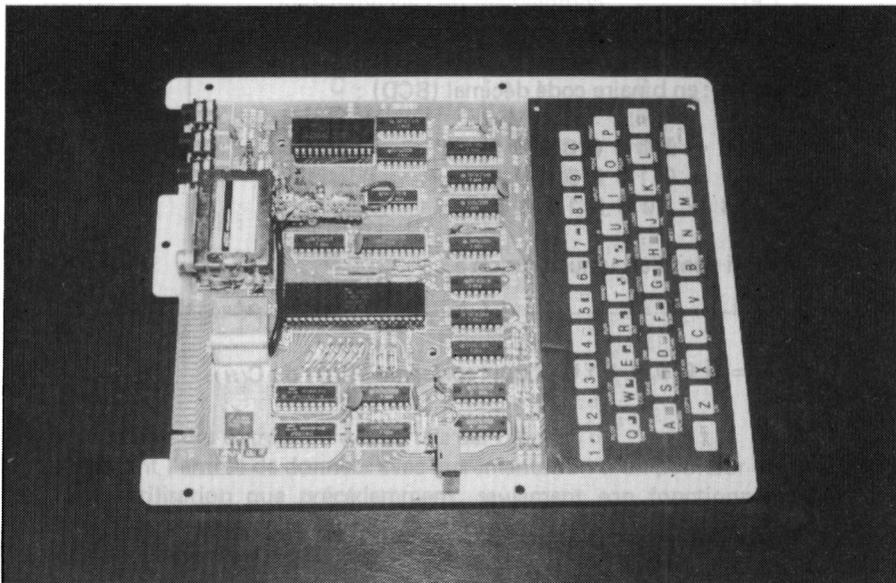
Après l'instruction DAA on aura :

A = 0 0 0 1 0 0 1 0 = 12 en BCD.

- PUSH : permet de sauvegarder 2 octets dans la pile. Ces 2 octets sont des registres pairs (BC, DE, HL, AF, IX, IY). Dès qu'une instruction PUSH est décodée, le pointeur de pile est d'abord décrémenté et le registre de poids fort (B,D,H...) est stocké dans la mémoire adressée. Ensuite, le pointeur est de nouveau décrémenté et le registre de poids faible (C, E, L...) est stocké dans la nouvelle mémoire adressée.
- POP : est l'opposée du PUSH. Cette instruction permet de récupérer des données dans la pile par groupe de 2 octets. Il est important de se rappeler que le dernier groupe de 2 octets entré est le premier à sortir.



ATTENTION : Il est important de se souvenir que si dans un programme il est fait appel à un sous-programme par un CALL, l'adresse de retour est sauvegardée également dans la pile et un POP mal placé aura pour effet de la faire sortir et ainsi de « planter » le programme. Il est donc préférable qu'un PUSH dans un programme n'entraîne pas un POP dans un sous-programme ou inversement sinon avec de grandes précautions.



LE ZX 80: TECHNOLOGIE PLUS CLASSIQUE.

DEUXIEME PARTIE

LES PROGRAMMES «UTILITAIRES»

- DESASSEMBLEUR**
- CREATION AUTOMATIQUE DE REM**
- TRANSFERT D'UN EPROM EN RAM**

LES PROGRAMMES A VOCATION RADIO

- TRANSMISSION D'IMAGES**
- EMISSION-RECEPTION MORSE**
- CALCUL QTH LOCATOR ET AZIMUT**
- SUIVI DE CONTEST**
- CALCULS DIVERS**
- MIRE**
- QSO TV**
- FICHER DEPARTEMENTAL**
- EMISSION-RECEPTION RTTY**
- FICHER TRI**

Dans la seconde partie de cet ouvrage qui, nous le rappelons, se voulait être non un cours magistral mais une présentation illustrée d'exemples, nous vous proposons quelques programmes commentés, permettant d'entrevoir quelques unes des applications possibles du ZX.

Vous trouverez donc des programmes BASIC, mais aussi, et c'est important, des programmes en langage machine qui, nous l'espérons, vous permettront d'entreprendre à votre tour ce type de programmation.

Vous pourrez constater que, lors de la programmation en langage machine, nous avons essayé de tirer parti des routines qui existent déjà en ROM. En faisant appel à celles-ci, nous évitons, dans bien des cas, de récrire des fonctions que l'on trouve dans la ROM, telles que le CLS (effacement de l'écran), la lecture du clavier, le SCROLL et le PRINT, pour ne citer qu'elles.

Il est donc bien pratique de connaître un peu la ROM et son organisation interne, notamment au niveau de l'implantation de quelques unes de ces routines. Dans ce but, nous vous fournissons un programme de désassemblage (désassembleur) qui, nous le pensons, vous facilitera la tâche lors de vos recherches, ainsi que (voir figure 14) une table résumée des adresses de début des routines qui seront le plus souvent utilisées en ROM. Nous avons exclu les routines de calcul en virgule flottante pour ne retenir que les routines les plus facilement utilisables.

Rappelons que, à sa mise sous tension, le Z 80 va chercher ses instructions en 0 0 0 0.

Il faut aussi savoir que le générateur de caractères est implanté à partir de 1E00 (H) (7680 décimal) et que, pour cette raison, il sera inutile d'aller désassembler ce qui se trouve au-dessus de cette adresse.

A propos du générateur de caractères, nous croyons bon de vous rappeler sa conception. Chaque caractère est composé d'une matrice de points, 8 x 8. Les 64 caractères sont rangés en ROM en fonction de leur code SINCLAIR (0 à 63). Le premier est donc l'espace et le dernier le Z. Ils sont représentés en ROM, chacun par 8 octets.

Voici, un petit programme qui, lancé par RUN, vous permettra, après avoir introduit un caractère, de découvrir son emplacement dans le générateur et son profil binaire, ou son tracé agrandi en suivant les instructions du REM 200.

```

S      SREM GENERATEUR DE CARACTERE
      10 INPUT A$
      20 PRINT AT 0,15;A$
      30 PRINT
      40 LET J=7680+(8*CODE A$)
      50 FOR I=J TO J+7
      60 PRINT I;TAB 10;PEEK I;TAB 1
S;
      70 LET VAL=PEEK I
      80 FOR K=0 TO 7
      90 LET BIN=VAL-2**(7-K)
     100 IF BIN>=0 THEN GOTO 130
     110 PRINT "0";
     120 GOTO 150
     130 PRINT "1";
     140 LET VAL=BIN
     150 NEXT K
     170 PRINT
     180 NEXT I
     185 REM
     190 REM
     200 REM "VOUS POUVEZ AUSSI REMPL
LACER LES LIGNES 110,130, PAR
     110 PRINT " ";
     130 PRINT " ";

```

Rappelons que le programme ne fonctionne que pour les caractères dont le code est compris entre 0 et 63, car eux seuls sont présents dans le générateur, l'inversion vidéo faisant appel à une routine particulière.

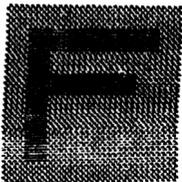
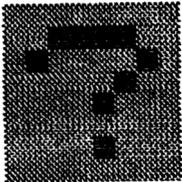
Voici quelques exemples d'exécution. Notez au passage que l'interligne (pour les chiffres et les lettres) est fait par le générateur (première et dernière ligne de la matrice à zéro), alors que pour les caractères graphiques, il n'y a pas d'interligne....

Figure 14

Table d'implantation des principales routines en ROM

0F23	FAST
0F2B	SLOW
02F6	SAVE
0340	LOAD
0A2A	CLS
0C0E	SCROLL
0CDC	STOP
0869	COPY
0F32	PAUSE
03C3	NEW
03E5	initialisations
02BB	lecture clavier
07BD	décodage d'une touche

Il faudra apprendre à les utiliser car toutes ne peuvent pas être appelées sans quelques initialisations préalables.

<pre> 000004 000005 000006 000007 000008 000009 000010 000011 000012 000013 000014 000015 000016 000017 000018 000019 000020 000021 000022 000023 000024 000025 000026 000027 000028 000029 000030 000031 </pre>	<pre> 0 126 54 124 54 54 54 54 54 0 </pre>		<pre> 7984 7985 7986 7987 7988 7989 7990 7991 </pre>	<pre> R 0 50 56 56 56 126 56 56 0 </pre>	<pre> 00000000 00111100 01000010 01000010 01111110 01000010 01000010 01000010 00000000 </pre>
<pre> 7800 7801 7802 7803 7804 7805 7806 7807 </pre>	<pre> 0 50 56 4 8 8 8 0 </pre>		<pre> 7728 7729 7730 7731 7732 7733 7734 7735 </pre>	<pre> 15 15 15 15 240 240 240 240 </pre>	<pre> 00001111 00001111 00001111 00001111 11110000 11110000 11110000 11110000 </pre>

DESASSEMBLEUR

Le premier travail pour entrer ce programme en machine est de créer les tables nécessaires à son fonctionnement. Ces tables sont contenues dans les variables A\$, B\$ et C\$.

Entrez dans un premier temps le petit programme suivant :

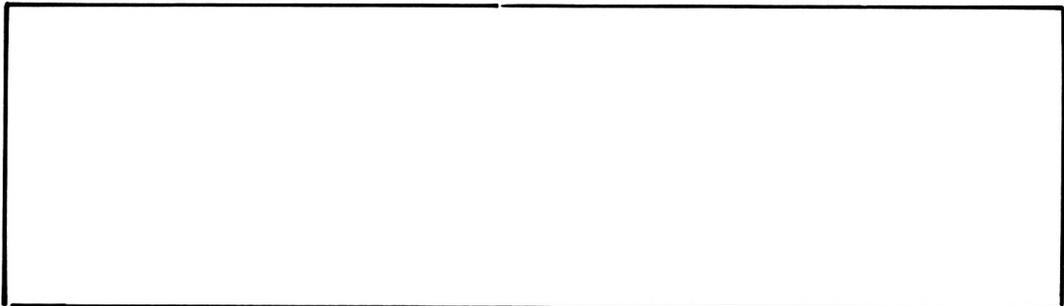
Il suffit de faire RUN.

Une fois les trois tables entrées, tapez le programme principal. Attention, une fois ce travail terminé, sauvegardez le tout sur K7 en faisant GOTO 1000 afin d'éviter toute fausse manipulation, telle que RUN, CLEAR qui aurait pour effet d'effacer les trois tables.

Un fois la sauvegarde terminée, le programme se lance automatiquement.

```
10 DIM A$(264,10)
20 DIM B$(264,10)
30 DIM C$(189,10)
40 SCROLL
50 PRINT TAB 10,"A$"
60 FOR N=1 TO 264
65 SCROLL
67 PRINT N;
70 INPUT A$(N)
80 PRINT "-";A$(N)
90 NEXT N
100 SCROLL
110 PRINT TAB 10;"B$"
120 FOR N=1 TO 264
130 SCROLL
135 PRINT N;
140 INPUT B$(N)
150 PRINT "-";B$(N)
160 NEXT N
170 SCROLL
180 PRINT TAB 10;"C$"
190 FOR N=65 TO 189
200 SCROLL
205 PRINT N;
210 INPUT C$(N)
220 PRINT "-";C$(N)
230 NEXT N
```

*Ce programme permettra d'entrer
les 3 tables A B\$ et C\$*



DESASSEMBLEUR

```

1 REM "DESAS"
5 CLS
0 PRINT "DONNEZ L ADRESSE EN
DECIMAL DU DEBUT DE LA ZONE A D
DESASSEMBLER"
10 INPUT A
15 GOSUB 200
200 CLS
205 FOR K=1 TO 22
300 PRINT A-W; "-";
400 LET O=PEEK A
500 IF O=203 THEN GOTO 500
600 IF O=207 THEN GOTO 550
700 IF O=221 OR O=253 THEN GOTO
600
900 GOTO 700
1000 STOP
200 PRINT "EST-CE L'ADRESSE RE
ELLE DE LA"
210 PRINT "ROUTINE (O/N)?"
220 INPUT R$
230 IF R$="O" THEN GOTO 300
240 PRINT "DONNEZ L'ADRESSE RE
ELLE ."
250 INPUT O
260 LET W=A-O
270 RETURN
300 LET W=0
310 RETURN
500 LET O=PEEK (A+1)
505 PRINT A$(O+1)
510 LET A=A+2
520 GOTO 200
550 LET O=PEEK (A+1)
555 PRINT C$(O+1);
560 IF O=67 OR O=75 OR O=83 OR
O=91 OR O=99 OR O=107 OR O=115 O
R
O=123 THEN GOTO 580
562 PRINT
565 LET A=A+2
570 GOTO 200
580 PRINT "NN=";PEEK (A+2)+PEEK
(A+3)*256
585 LET A=A+4
590 GOTO 200
600 PRINT B$(O+1);
601 PRINT PEEK A;"/";
605 PRINT PEEK (A+1);"/";
606 LET AD=A+2
610 LET O=PEEK (A+1)
615 IF O=33 OR O=34 OR O=42 OR
O=54 OR O=203 THEN GOTO 630
620 IF O=52 OR O=53 OR O=70 OR
O=78 OR O=86 OR O=94 OR O=102 OR
O=110 OR O=112 OR O=113 OR O=11
4 OR O=115 OR O=116 OR O=117 OR
O=119 OR O=126 OR O=134 OR O=142
OR O=150 OR O=158 OR O=166 OR O
=174 OR O=182 OR O=190 THEN GOTO

```

A contient l'adresse de début de la zone mémoire à désassembler. Compteur de lignes à afficher. Affiche la 1ère adresse.

Test si le 1er octet de l'instruction est en hexa : CB ; ED ; DD ou FD, sinon GOTO 700.

Sous-programme permettant de connaître l'adresse réelle de la routine si elle a été déplacée avant le chargement du désassembleur.

w = ad du désassemblage - ad. réelle
w = 0 si la routine est à sa place.

CAS où le 1er octet est CB : recherche instruction dans table A\$ et retour CAS où le 1er octet est ED : Recherche dans C\$ + lecture si nécessaire des deux octets suivants + retour.

CAS où le 1er octet est DD ou FD :

Il s'agit d'instructions sur IX ou IY (non décodées). Les codes décimaux suivants sont affichés (1 ou 2 suivant le cas).

```

0000 GOTO 660
0005 PRINT PEEK (A+2);"/";
0010 PRINT PEEK (A+3)
0015 LET AD=A+4
0020 GOTO 660
0025 PRINT PEEK (A+2)
0030 LET AD=A+3
0035 LET A=AD
0040 GOTO 2000
0045 PRINT B$(O+1);
0050 IF O=219 OR O=6 OR O=14 OR
O=22 OR O=30 OR O=38 OR O=46 OR
O=54 OR O=62 OR O=198 OR O=206 O
R O=211 OR O=214 OR O=222 OR O=2
30 OR O=238 OR O=246 OR O=254 TH
EN GOTO 750

0070 IF O=1 OR O=17 OR O=33 OR O
=34 OR O=42 OR O=49 OR O=50 OR O
=56 OR O=194 OR O=195 OR O=196 O
R O=202 OR O=204 OR O=205 OR O=2
10 OR O=212 OR O=218 OR O=220 OR
O=226 OR O=228 OR O=234 OR O=23
6 OR O=242 OR O=244 OR O=250 OR
O=252 THEN GOTO 760
0075 IF O=16 OR O=24 OR O=32 OR
O=40 OR O=48 OR O=56 THEN GOTO 8
00

0080 PRINT
0085 LET A=A+1
0090 GOTO 2000
0095 PRINT "N=";PEEK (A+1)
0100 LET A=A+2
0105 GOTO 2000
0110 PRINT "NN="; (PEEK (A+1)+PEE
K (A+2)*256)
0115 LET A=A+3
0120 GOTO 2000
0125 PRINT "D=";PEEK (A+1);" ";
0130 LET D=PEEK (A+1)
0135 IF D>127 THEN LET D=D-256
0140 LET JR=A-U+2+D
0145 PRINT "(";JR;")"
0150 LET A=A+2
0155 GOTO 2000
0160 SAVE "DESAS"
0165 GOTO 5
0170 NEXT K
0175 IF INKEY$="" THEN GOTO 2020
0180 IF INKEY$="Z" THEN COPY
0185 IF INKEY$="N" THEN GOTO 5
0190 CLS
0195 GOTO 25

```

1-RLC B	2-RLC C
3-RLC D	4-RLC E
5-RLC H	6-RLC L
7-RLC (HL)	8-RLC A
9-RRC B	10-RRC C
11-RRC D	12-RRC E
13-RRC H	14-RRC L
15-RRC (HL)	16-RRC A

CAS où l'instruction n'a pas de préfixe : Recherche dans B\$ + affiche des 1 ou 2 octets suivants selon le cas.

Test si sauts relatifs. Si oui, GOTO 800.

A la ligne 1000, le dernier S de «DESAS» doit être tapé sur le clavier en mode normal et non en mode graphique. L'inversion est faite automatiquement par l'ordinateur lors de la première sauvegarde.

Affichage valeur des sauts relatifs et calcul adresse.

Incrémente compteur lignes.

Test appui touche :
Z = imprime page sur imprimante
N = recommence avec nouvelle adresse
C = continue.

Variable A\$

117	-RRL	B		
119	-RRL	D		
121	-RRL	H		
123	-RRL	(HL)		
125	-RRR	B		
127	-RRR	D		
129	-RRR	H		
131	-RRR	(HL)		
133	-SLDA	B		
135	-SLDA	D		
137	-SLDA	H		
139	-SLDA	(HL)		
141	-SPRA	B		
143	-SPRA	D		
145	-SPRA	H		
147	-SPRA	(HL)		
149	-			
151	-			
153	-			
155	-			
157	-S	B		
159	-SRL	D		
161	-SRL	H		
163	-SRL	(HL)		
165	-BIT	B		
167	-BIT	D		
169	-BIT	H		
171	-BIT	(HL)		
173	-BIT	1,B		
175	-BIT	1,D		
177	-BIT	1,H		
179	-BIT	1,(HL)		
181	-BIT	2,B		
183	-BIT	2,D		
185	-BIT	2,H		
187	-BIT	2,(HL)		
189	-BIT	3,B		
191	-BIT	3,D		
193	-BIT	3,H		
195	-BIT	3,(HL)		
197	-BIT	4,B		
199	-BIT	4,D		
201	-BIT	4,H		
203	-BIT	4,(HL)		
205	-BIT	5,B		
207	-BIT	5,D		
209	-BIT	5,H		
211	-BIT	5,(HL)		
213	-BIT	6,B		
215	-BIT	6,D		
217	-BIT	6,H		
219	-BIT	6,(HL)		
221	-BIT	7,B		
223	-BIT	7,D		
225	-BIT	7,H		
227	-BIT	7,(HL)		
229	-RES	B		
231	-RES	D		
233	-RES	H		
235	-RES	(HL)		
237	-RES	1,B		
239	-RES	1,D		
241	-RES	1,H		
243	-RES	1,(HL)		
245	-RES	2,B		
247	-RES	2,D		
249	-RES	2,H		
251	-RES	2,(HL)		
253	-RES	3,B		
255	-RES	3,D		
257	-RES	3,H		
259	-RES	3,(HL)		
261	-RES	4,B		
263	-RES	4,D		
265	-RES	4,H		
267	-RES	4,(HL)		
269	-RES	5,B		
271	-RES	5,D		
273	-RES	5,H		
275	-RES	5,(HL)		
277	-RES	6,B		
279	-RES	6,D		
281	-RES	6,H		
283	-RES	6,(HL)		
285	-RES	7,B		
287	-RES	7,D		
289	-RES	7,H		
291	-RES	7,(HL)		
293	-RES	8,B		
295	-RES	8,D		
297	-RES	8,H		
299	-RES	8,(HL)		
301	-RES	9,B		
303	-RES	9,D		
305	-RES	9,H		
307	-RES	9,(HL)		
309	-RES	10,B		
311	-RES	10,D		
313	-RES	10,H		
315	-RES	10,(HL)		
317	-RES	11,B		
319	-RES	11,D		
321	-RES	11,H		
323	-RES	11,(HL)		
325	-RES	12,B		
327	-RES	12,D		
329	-RES	12,H		
331	-RES	12,(HL)		
333	-RES	13,B		
335	-RES	13,D		
337	-RES	13,H		
339	-RES	13,(HL)		
341	-RES	14,B		
343	-RES	14,D		
345	-RES	14,H		
347	-RES	14,(HL)		
349	-RES	15,B		
351	-RES	15,D		
353	-RES	15,H		
355	-RES	15,(HL)		
357	-RES	16,B		
359	-RES	16,D		
361	-RES	16,H		
363	-RES	16,(HL)		
365	-RES	17,B		
367	-RES	17,D		
369	-RES	17,H		
371	-RES	17,(HL)		
373	-RES	18,B		
375	-RES	18,D		
377	-RES	18,H		
379	-RES	18,(HL)		
381	-RES	19,B		
383	-RES	19,D		
385	-RES	19,H		
387	-RES	19,(HL)		
389	-RES	20,B		
391	-RES	20,D		
393	-RES	20,H		
395	-RES	20,(HL)		
397	-RES	21,B		
399	-RES	21,D		
401	-RES	21,H		
403	-RES	21,(HL)		
405	-RES	22,B		
407	-RES	22,D		
409	-RES	22,H		
411	-RES	22,(HL)		
413	-RES	23,B		
415	-RES	23,D		
417	-RES	23,H		
419	-RES	23,(HL)		
421	-RES	24,B		
423	-RES	24,D		
425	-RES	24,H		
427	-RES	24,(HL)		
429	-RES	25,B		
431	-RES	25,D		
433	-RES	25,H		
435	-RES	25,(HL)		
437	-RES	26,B		
439	-RES	26,D		
441	-RES	26,H		
443	-RES	26,(HL)		
445	-RES	27,B		
447	-RES	27,D		
449	-RES	27,H		
451	-RES	27,(HL)		
453	-RES	28,B		
455	-RES	28,D		
457	-RES	28,H		
459	-RES	28,(HL)		
461	-RES	29,B		
463	-RES	29,D		
465	-RES	29,H		
467	-RES	29,(HL)		
469	-RES	30,B		
471	-RES	30,D		
473	-RES	30,H		
475	-RES	30,(HL)		
477	-RES	31,B		
479	-RES	31,D		
481	-RES	31,H		
483	-RES	31,(HL)		
485	-RES	32,B		
487	-RES	32,D		
489	-RES	32,H		
491	-RES	32,(HL)		
493	-RES	33,B		
495	-RES	33,D		
497	-RES	33,H		
499	-RES	33,(HL)		
501	-RES	34,B		
503	-RES	34,D		
505	-RES	34,H		
507	-RES	34,(HL)		
509	-RES	35,B		
511	-RES	35,D		
513	-RES	35,H		
515	-RES	35,(HL)		
517	-RES	36,B		
519	-RES	36,D		
521	-RES	36,H		
523	-RES	36,(HL)		
525	-RES	37,B		
527	-RES	37,D		
529	-RES	37,H		
531	-RES	37,(HL)		
533	-RES	38,B		
535	-RES	38,D		
537	-RES	38,H		
539	-RES	38,(HL)		
541	-RES	39,B		
543	-RES	39,D		
545	-RES	39,H		
547	-RES	39,(HL)		
549	-RES	40,B		
551	-RES	40,D		
553	-RES	40,H		
555	-RES	40,(HL)		
557	-RES	41,B		
559	-RES	41,D		
561	-RES	41,H		
563	-RES	41,(HL)		
565	-RES	42,B		
567	-RES	42,D		
569	-RES	42,H		
571	-RES	42,(HL)		
573	-RES	43,B		
575	-RES	43,D		
577	-RES	43,H		
579	-RES	43,(HL)		
581	-RES	44,B		
583	-RES	44,D		
585	-RES	44,H		
587	-RES	44,(HL)		
589	-RES	45,B		
591	-RES	45,D		
593	-RES	45,H		
595	-RES	45,(HL)		
597	-RES	46,B		
599	-RES	46,D		
601	-RES	46,H		
603	-RES	46,(HL)		
605	-RES	47,B		
607	-RES	47,D		
609	-RES	47,H		
611	-RES	47,(HL)		
613	-RES	48,B		
615	-RES	48,D		
617	-RES	48,H		
619	-RES	48,(HL)		
621	-RES	49,B		
623	-RES	49,D		
625	-RES	49,H		
627	-RES	49,(HL)		
629	-RES	50,B		
631	-RES	50,D		
633	-RES	50,H		
635	-RES	50,(HL)		
637	-RES	51,B		
639	-RES	51,D		
641	-RES	51,H		
643	-RES	51,(HL)		
645	-RES	52,B		
647	-RES	52,D		
649	-RES	52,H		
651	-RES	52,(HL)		
653	-RES	53,B		
655	-RES	53,D		
657	-RES	53,H		
659	-RES	53,(HL)		
661	-RES	54,B		
663	-RES	54,D		
665	-RES	54,H		
667	-RES	54,(HL)		
669	-RES	55,B		
671	-RES	55,D		
673	-RES	55,H		
675	-RES	55,(HL)		
677	-RES	56,B		
679	-RES	56,D		
681	-RES	56,H		
683	-RES	56,(HL)		
685	-RES	57,B		
687	-RES	57,D		
689	-RES	57,H		
691	-RES	57,(HL)		
693	-RES	58,B		
695	-RES	58,D		
697	-RES	58,H		
699	-RES	58,(HL)		
701	-RES	59,B		
703	-RES	59,D		
705	-RES	59,H		
707	-RES	59,(HL)		
709	-RES	60,B		
711	-RES	60,D		
713	-RES	60,H		
715	-RES	60,(HL)		
717	-RES	61,B		
719	-RES	61,D		
721	-RES	61,H		
723	-RES	61,(HL)		
725	-RES	62,B		
727	-RES	62,D		
729	-RES	62,H		
731	-RES	62,(HL)		
733	-RES	63,B		
735	-RES	63,D		
737	-RES	63,H		
739	-RES	63,(HL)		
741	-RES	64,B		
743	-RES	64,D		
745	-RES	64,H		
747	-RES	64,(HL)		
749	-RES	65,B		
751	-RES	65,D		
753	-RES	65,H		
755	-RES	65,(HL)		
757	-RES	66,B		
759	-RES	66,D		
761	-RES	66,H		
763	-RES	66,(HL)		
765	-RES	67,B		
767	-RES	67,D		
769	-RES	67,H		
771	-RES	67,(HL)		
773	-RES	68,B		
775	-RES	68,D		
777	-RES	68,H		
779	-RES	68,(HL)		
781	-RES	69,B		
783	-RES	69,D		
785	-RES	69,H		
787	-RES	69,(HL)		
789	-RES	70,B		
791	-RES	70,D		
793	-RES	70,H		
795	-RES	70,(HL)		
797	-RES	71,B		
799	-RES	71,D		
801	-RES	71,H		
803	-RES	71,(HL)		
805	-RES	72,B		
807	-RES	72,D		
809	-RES	72,H		
811	-RES	72,(HL)		
813	-RES	73,B		
815	-RES	73,D		
817	-RES	73,H		
819	-RES	73,(HL)		
821	-RES	74,B		
823	-RES	74,D		
825	-RES			

147	-R	2	D	148	-R	2	D
149	-R	2	H	150	-R	2	H
151	-R	3	(HL)	152	-R	3	(HL)
153	-R	3	B	154	-R	3	B
155	-R	3	D	156	-R	3	D
157	-R	3	H	158	-R	3	H
159	-R	3	(HL)	160	-R	3	(HL)
161	-R	4	B	162	-R	4	B
163	-R	4	D	164	-R	4	D
165	-R	4	H	166	-R	4	H
167	-R	4	(HL)	168	-R	4	(HL)
169	-R	5	B	170	-R	5	B
171	-R	5	D	172	-R	5	D
173	-R	5	H	174	-R	5	H
175	-R	5	(HL)	176	-R	5	(HL)
177	-R	6	B	178	-R	6	B
179	-R	6	D	180	-R	6	D
181	-R	6	H	182	-R	6	H
183	-R	6	(HL)	184	-R	6	(HL)
185	-R	7	B	186	-R	7	B
187	-R	7	D	188	-R	7	D
189	-R	7	H	190	-R	7	H
191	-R	7	(HL)	192	-R	7	(HL)
193	-R	8	B	194	-R	8	B
195	-R	8	D	196	-R	8	D
197	-R	8	H	198	-R	8	H
199	-R	8	(HL)	200	-R	8	(HL)
201	-R	1	B	202	-R	1	B
203	-R	1	D	204	-R	1	D
205	-R	1	H	206	-R	1	H
207	-R	1	(HL)	208	-R	1	(HL)
209	-R	2	B	210	-R	2	B
211	-R	2	D	212	-R	2	D
213	-R	2	H	214	-R	2	H
215	-R	2	(HL)	216	-R	2	(HL)
217	-R	3	B	218	-R	3	B
219	-R	3	D	220	-R	3	D
221	-R	3	H	222	-R	3	H
223	-R	3	(HL)	224	-R	3	(HL)
225	-R	4	B	226	-R	4	B
227	-R	4	D	228	-R	4	D
229	-R	4	H	230	-R	4	H
231	-R	4	(HL)	232	-R	4	(HL)
233	-R	5	B	234	-R	5	B
235	-R	5	D	236	-R	5	D
237	-R	5	H	238	-R	5	H
239	-R	5	(HL)	240	-R	5	(HL)
241	-R	6	B	242	-R	6	B
243	-R	6	D	244	-R	6	D
245	-R	6	H	246	-R	6	H
247	-R	6	(HL)	248	-R	6	(HL)
249	-R	7	B	250	-R	7	B
251	-R	7	D	252	-R	7	D
253	-R	7	H	254	-R	7	H
255	-R	7	(HL)	256	-R	7	(HL)
257	-R			258	-R		
259	-R			260	-R		
261	-R			262	-R		
263	-R			264	-R		

```

1-NOP
3-LD (BC),A
5-INC B,B
7-LD B,N
9-EX AF,AF"
11-LD A,(BC)
13-INC C
15-LD C,N
17-DUNZ D
19-LD (DE),A
21-INC D
23-LD D,N
25-LR D,N
27-LD A,(DE)
29-INC E
31-LR E,N
33-LRNZ D
35-LD (NN),HL
37-INC H
39-LD H,N
41-LRZ D
43-LD HL,(NN)
45-INC L
47-LD L,N
49-LRNC D
51-LD (NN),A
53-INC (HL)
55-LD (HL),N
57-LRZ D
59-LD A,(NN)
61-INC A
63-LD A,N
65-LD B,B
67-LD B,D
69-LD B,H
71-LD B,(HL)
73-LD C,B
75-LD C,D
77-LD C,H
79-LD C,(HL)
81-LD D,B
83-LD D,D
85-LD D,H
87-LD D,(HL)
89-LD E,B
91-LD E,D
93-LD E,H
95-LD E,(HL)
97-LD H,B
99-LD H,D
101-LD H,H
103-LD H,(HL)
105-LD L,B
107-LD L,D
109-LD L,H
111-LD L,(HL)
113-LD (HL),B
115-LD (HL),D
117-LD (HL),H
119-HALT
121-LD A,B
123-LD A,D
125-LD A,H
127-LD A,(HL)
2-LD BC,NN
4-INC BC
6-DEC B
8-RLCA
10-ADD HL,BC
12-DEC BC
14-DEC C
16-RRCA
18-LD DE,NN
20-INC DE
22-DEC D
24-RLA
26-ADD HL,DE
28-DEC DE
30-DEC E
32-RRR
34-LD HL,NN
36-INC HL
38-DEC H
40-DAR
42-ADD HL,HL
44-DEC HL
46-DEC L
48-CPL
50-LD SP,NN
52-INC SP
54-DEC (HL)
56-SCF
58-ADD HL,SP
60-DEC SP
62-DEC A
64-CCF
66-LD B,C
68-LD B,E
70-LD B,(C)
72-LD B,(E)
74-LD C,(C)
76-LD C,(E)
78-LD C,(A)
80-LD C,(E)
82-LD D,(C)
84-LD D,(E)
86-LD D,(A)
88-LD D,(E)
90-LD E,(C)
92-LD E,(E)
94-LD E,(A)
96-LD E,(E)
98-LD H,(C)
100-LD H,(E)
102-LD H,(A)
104-LD H,(E)
106-LD L,(C)
108-LD L,(E)
110-LD L,(A)
112-LD L,(E)
114-LD (HL),(C)
116-LD (HL),(E)
118-LD (HL),(A)
120-LD (HL),(E)
122-LD A,(C)
124-LD A,(E)
126-LD A,(A)
128-LD A,(A)

```

139	-ADD	A,B	139	-ADD	A,C
139A	-ADD	D,V	139B	-ADD	A,FM
139B	-ADD	A,H	139C	-ADD	A,FL
139D	-ADD	A,(HL)	139D	-ADD	A,FL
139E	-ADD	A,B	139E	-ADD	A,FL
139F	-ADD	A,V	139F	-ADD	A,FL
140	-ADD	A,H	140	-ADD	A,FL
140A	-ADD	A,(HL)	140A	-ADD	A,FL
140B	-SUB	B	140B	-SUB	C
140C	-SUB	D	140C	-SUB	FM
140D	-SUB	H	140D	-SUB	FL
140E	-SUB	(HL)	140E	-SUB	FL
140F	-SBC	A,B	140F	-SBC	A,FM
141	-SBC	A,V	141	-SBC	A,FM
141A	-SBC	A,H	141A	-SBC	A,FM
141B	-SBC	A,(HL)	141B	-SBC	A,FM
141C	-AND	B	141C	-AND	C
141D	-AND	D	141D	-AND	FM
141E	-AND	H	141E	-AND	FL
141F	-AND	(HL)	141F	-AND	FL
142	-XOR	B	142	-XOR	C
142A	-XOR	D	142A	-XOR	FM
142B	-XOR	H	142B	-XOR	FL
142C	-XOR	(HL)	142C	-XOR	A
142D	-OR	B	142D	-OR	C
142E	-OR	D	142E	-OR	FM
142F	-OR	H	142F	-OR	FL
143	-OR	(HL)	143	-OR	FL
143A	-CP	B	143A	-CP	C
143B	-CP	D	143B	-CP	FM
143C	-CP	H	143C	-CP	FL
143D	-CP	(HL)	143D	-CP	FL
143E	-RET	NZ	143E	-POP	BC
143F	-JPNZ	NN	143F	-JP	NN
144	-CALL	NZ,NN	144	-PUSH	BC
144A	-ADD	A,N	144A	-RST	0
144B	-RET	Z	144B	-RET	
144C	-JPZ	NN	144C	-	
144D	-CALL	Z,NN	144D	-CALL	NN
144E	-ADC	A,N	144E	-RST	5
144F	-RET	NC	144F	-POP	DE
145	-JPNC	NN	145	-OUT	N,A
145A	-CALL	NC,NN	145A	-PUSH	DE
145B	-SUB	Z	145B	-RST	10
145C	-RET	C	145C	-EXX	
145D	-JPC	NN	145D	-IN	A,N
145E	-CALL	C,NN	145E	-INST	/IX
145F	-SBC	A,N	145F	-RST	24
146	-RET	PO	146	-POP	HL
146A	-JP	PO,NN	146A	-EX	(SP),HL
146B	-CALL	PO,NN	146B	-PUSH	HL
146C	-AND	Z	146C	-RST	30
146D	-RET	PE	146D	-JP	(HL)
146E	-JP	PE,NN	146E	-EX	DE,HL
146F	-CALL	PE,NN	146F	-	
147	-XOR	Z	147	-RST	40
147A	-RET	P	147A	-POP	AF
147B	-JP	P,NN	147B	-DI	
147C	-CALL	P,NN	147C	-PUSH	AF
147D	-OR	Z	147D	-RST	48
147E	-RET	M	147E	-LD	SP,HL
147F	-JP	M,NN	147F	-EI	
148	-CALL	M,NN	148	-INST	/IY
148A	-CP	Z	148A	-RST	56

00000000 -
00000001 -
00000002 -
00000003 -

00000000 -
00000001 -
00000002 -
00000003 -

variable C\$

008 - IN B, (C)	008 - OUT (C), B
009 - SBC HL, BC	009 - LD (NN), BC
010 - NEG	010 - RETN
011 - IM 0	011 - LD I, A
013 - IN C, (C)	013 - OUT (C), C
015 - ADC HL, BC	015 - LD BC, (NN)
017 -	017 - RETI
019 -	019 - LD R, A
021 - IN D, (C)	021 - OUT (C), D
023 - SBC HL, DE	023 - LD (NN), DE
025 -	025 -
027 - IM 1	027 - LD A, I
029 - IN E, (C)	029 - OUT (C), E
031 - ADC HL, DE	031 - LD DE, (NN)
033 -	033 -
035 - IM 2	035 - LD A, R
037 - IN H, (C)	037 - OUT (C), H
039 - SBC HL, HL	039 - LD (NN), HL
041 -	041 -
043 -	043 -
045 - IN L, (C)	045 - RRD
047 - ADC HL, HL	047 - OUT (C), L
049 -	049 - LD DE, (NN)
051 -	051 -
053 -	053 -
055 - SBC HL, SP	055 - RLD
057 -	057 -
059 -	059 -
061 -	061 - LD (NN), SP
063 -	063 -
065 -	065 -
067 -	067 -
069 -	069 -
071 - IN A, (C)	071 - OUT (C), A
073 - ADC HL, SP	073 - LD SP, (NN)
075 -	075 -
077 -	077 -
079 -	079 -
081 -	081 -
083 -	083 -
085 -	085 -
087 -	087 -
089 -	089 -
091 -	091 -
093 -	093 -
095 -	095 -
097 -	097 -
099 -	099 -
101 -	101 -
103 -	103 -
105 -	105 -
107 -	107 -
109 -	109 -
111 -	111 -
113 -	113 -
115 -	115 -
117 -	117 -
119 -	119 -
121 -	121 -
123 -	123 -
125 -	125 -
127 -	127 -
129 -	129 -
131 -	131 -
133 -	133 -
135 -	135 -
137 -	137 -
139 -	139 -
141 -	141 -
143 -	143 -
145 -	145 -
147 -	147 -
149 -	149 -
151 -	151 -
153 -	153 -
155 -	155 -
157 -	157 -
159 -	159 -
161 - LDI	161 -
163 - INI	163 - CPI
165 -	165 - OUTI
167 -	167 -
169 - LDD	169 -
171 - IND	171 - CPD
173 -	173 - OUTD
175 -	175 -
177 - LDIR	177 -
179 - INIR	179 - CPIR
	180 - OTIR

181-
183-
185-LDDR
187-INDR
189-

182-
184-
186-CPDR
188-OTDR

Mode d'emploi du programme «DESAS»

Dès son lancement il vous demande l'adresse de début de la routine à désassembler (en décimal). Puis, il demande si cette adresse est l'adresse réelle. En effet, si vous voulez désassembler une routine dans la ROM, par exemple à l'adresse 971, cette adresse est réelle. Par contre, en supposant que vous disposiez d'un programme en langage machine qui se trouve dans un REM et que vous vouliez le désassembler, vous allez être obligé de transférer ce programme dans un espace mémoire protégé avant de charger le désassembleur (au-dessus de RAMTOP). Seulement l'adresse où vous allez ensuite lancer le décodage n'est pas l'adresse réelle, aussi, le programme dispose d'une fonction qui permet avant l'affichage de calculer la bonne adresse.

Exemple : la routine à désassembler est dans un REM qui s'étend de 16514 à 17000. La première chose à faire est de réserver de la place au-dessus de RAMTOP. Dans notre cas il faut :

$$17000 - 16514 = 486, \text{ soit } 486 + 1 = 487 \text{ octets.}$$

Nous devons donc mettre RAMTOP à $32768 - 487 = 32281$, soit $126 \times 256 + 25$.

Il suffit de faire :
POKE 16389, 126
POKE 16388, 25
NEW

Le programme est ensuite chargé et transféré ainsi :

```
LET N = 32281
FOR L = 16514 TO 17000
POKE N, PEEK L
LET N = N + 1
NEXT N
```

Puis on charge le désassembleur :

Adresse de début : 32281

Adresse réelle ? Non

Adresse réelle = 16514

Et le désassemblage commence.

REM (création automatique de REM)

Ce programme en langage machine permet de créer en ligne No 1 du basic un REM de longueur choisie et permettant ainsi de disposer d'un espace de 0 à 5000 octets pour des routines en code machine.

Le programme se lance automatiquement après son chargement et vous demande le nombre d'octets que vous voulez dans le REM (0 à 5000). Après votre réponse, un REM va se créer en ligne No 1, de la longueur que vous avez demandée et rempli du caractère espace inversé. Ensuite, le ZX 81 va se réinitialiser et effacer toute sa mémoire sauf bien sûr le REM. Il ne reste plus qu'à écrire votre programme basic à partir de la ligne 2 ou de rentrer vos codes machine dans le REM ainsi créé.

ATTENTION : RAPPEL : Si dans un listing, le REM en ligne No 1 est d'une longueur supérieure à la capacité de l'écran, il ne faut en aucun cas effacer la ligne qui suit ce REM (par exemple : la ligne No 2), sous peine de perdre le contrôle du ZX 81 et de ce fait de couper et de remettre l'alimentation, d'où perte du programme.

Listing du programme : (ne rien modifier ou ajouter, sinon « plantage »)

```

5 DIM B$(4000)
10 LET A$="5"+CHR$ 125+"RND0 7
0 7770"+CHR$ 234+"5 )RND6"+CHR
$ 127+"RND"+CHR$ 229+"."0"+CHR$ 1
13+CHR$ 209+CHR$ 229+"5"+CHR$ 13
0+"RND:.""+CHR$ 109+CHR$ 167+"4."
+CHR$ 122+CHR$ 107+"C#0"/"+CHR$
242+CHR$ 225+"7"+CHR$ 195+"#."
20 PRINT "NB D" "OCTETS DANS LE
REM(5000MAX) ?"
30 INPUT N
40 LET N=N+2
50 LET L=21767
60 POKE (L+15),INT (N/256)
70 POKE (L+14),N-(PEEK (L+15) *
256)
80 RAND USR L
100 CLEAR
101 SAVE "REM"
110 RUN

```

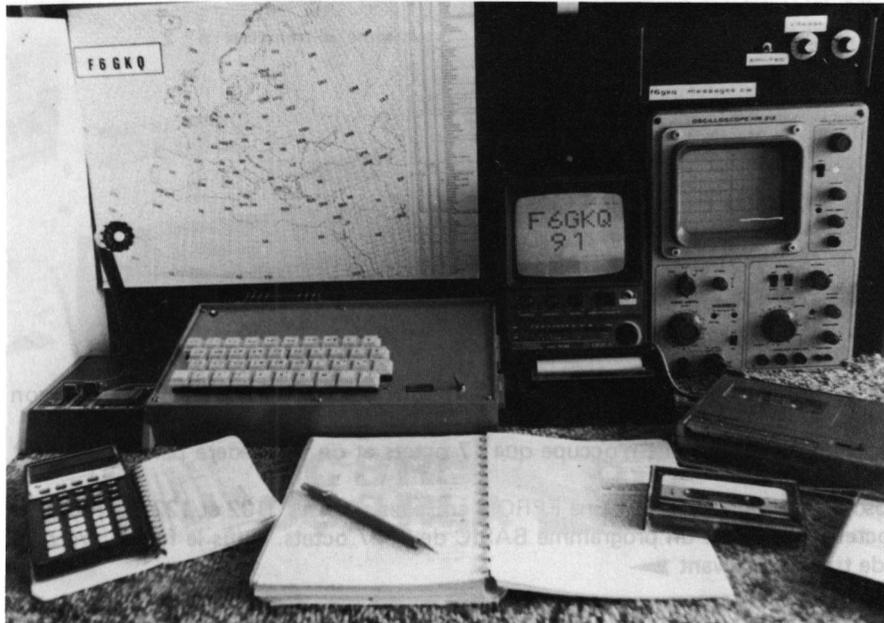
Sauvegarder sur K7 par
GOTO 100

1 REM

Exemple : 150 octets

DESASSEMBLAGE ROUTINE REM

<pre> 01767-LD HL,NN NN=16509 01770-LD (HL),N N=0 01772-INC HL 01773-LD (HL),N N=1 01775-INC HL 01776-INC HL 01777-INC HL 01778-LD (HL),N N=234 01780-LD HL,NN NN=0 01783-LD DE,NN NN=16512 01786-LD (NN),HL NN=16511 01789-PUSH HL 01790-ADD HL,DE 01791-LD (HL),N N=118 01793-POP DE 01794-PUSH HL 01795-LD HL,NN NN=16514 01798-DEC DE 01799-DEC DE 01800-LD A,E 01801-AND A 01802-JR NZ,D D=4 (21808) 01804-LD A,D 01805-AND A 01806-JR Z,D D=5 (21813) 01808-LD (HL),N N=126 01810-INC HL 01811-JR D D=242 (21799) 01813-POP HL 01814-INC HL 01815-JP NN NN=1027 01818-NOP 01819-</pre>	<div style="border-left: 1px solid black; border-right: 1px solid black; padding: 0 10px;"> <p>Affiche le numéro de la ligne du REM : ligne n° 1</p> </div> <p>— 234 = code de «REM» — HL contient le nombre d'octets du REM +1</p> <p>— Met 118 à la fin du REM — DE contient le nombre d'oc- tets +1 — HL pointe le début du REM</p> <div style="border-left: 1px solid black; border-right: 1px solid black; padding: 0 10px;"> <p>Boucle de remplissage du REM avec le code 128 (espace inversé)</p> </div> <p>— Récupère adresse de fin du REM +118 — Saut à la routine d'initiali- sation dans la ROM</p>
---	---



L'ENSEMBLE DE TRAVAIL AU COMPLET.

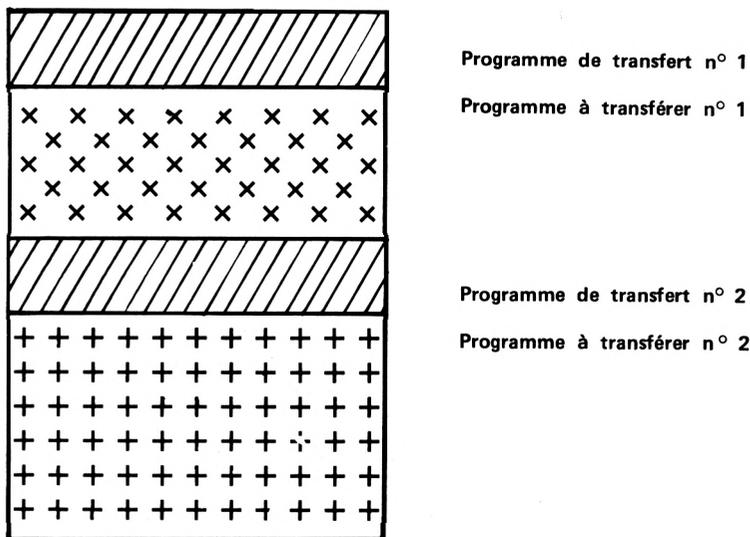
TRANSFERT D'UN PROGRAMME BASIC D'EPROM EN RAM

Un des problèmes du ZX est, nous l'avons vu, le temps de chargement des programmes à partir d'un magnétophone à cassettes. Certains d'entre vous seront peut-être tentés d'implanter en EPROM les programmes qu'ils utilisent le plus souvent, surtout s'ils ne sont pas trop longs. Cela peut se concevoir pour le QTH LOCATOR, le programme ERCW et bien d'autres qui tiennent facilement en EPROM.

Le problème est qu'on ne peut exécuter des programmes BASIC que s'ils sont implantés à partir de 16509. Il faudra donc faire un petit programme de transfert de l'EPROM vers la RAM.

Ce programme tient en quelques octets et il faudra l'écrire devant chaque programme résidant en EPROM et devant être transféré.

On obtient l'architecture suivante :



Chaque programme de transfert contient les paramètres du programme à transférer (son adresse, le nombre d'octets, etc...).

Un programme de transfert n'occupe que 17 octets et on y accédera par RAND USR adresse.

Supposons que nous ayons logé une EPROM entre les adresses 8192 et 12288 (2000H et 3000H) de 4 K octets, contenant un programme BASIC de 3527 octets. Nous le ferons précéder du programme de transfert suivant ►

8192	LD HL, 8209	33, 17, 32	(adresse du programme en EPROM)
8195	LD DE, 16509	17, 125, 64	(début de zone programme)
8198	LD BC, 3527	1, 199, 13	(nombre d'octets à transférer)
8201	LDIR	237, 187	(opération de transfert)
8203	LD HL, 20037	33, 69, 78	(16509 + nb d'octets à transférer + 1)
8206	JP 1027	195, 3, 4	et saut à routine initialisation
8209	début du programme		

On accédera au programme par RAND USR 8192.

Il se trouve chargé instantanément. C'est bien pratique !

Nous remarquons ici l'utilisation d'une instruction très puissante du Z 80 LDIR qui effectue un transfert d'un bloc de données (ici nos 3527 octets) d'un endroit de la mémoire pointé par HL (ici 8209) à un autre, pointé par DE (ici 16509), le tout en un temps record.

Chaque programme BASIC résidant en EPROM sera donc précédé d'un tel programme où on aura modifié les adresses et nombres d'octets.



PROGRAMME DE TRANSMISSION D'IMAGES ---

Une petite mise en garde d'abord. Ce programme appelé «TVBV» permet de transmettre par tout ou rien une image élaborée sur un ZX 81, avec une très basse définition, vers un autre ZX 81. Ceci n'est pas de la SSTV, mais une approche avec les seuls «moyens du bord» qui, dans le cas de pas mal de radioamateurs, se limitent à notre petit Sinclair. Cependant, ce principe de transmission n'ayant rien de comparable avec ceux existants déjà et autorisés en radio, il ne devra être utilisé qu'en local et non via émetteur—récepteur.

Ce programme qui a donc comme but la démonstration des possibilités du ZX 81, est composé de deux parties.

La première permet de dessiner à l'écran via le clavier avec comme définition, non pas le plot, mais le caractère, c'est à dire $24 \times 32 = 768$ points. C'est vraiment de la très basse définition, mais avec un peu de doigté et beaucoup d'imagination, il est possible de faire de belles choses. Le principe de fonctionnement de cette partie du programme réside sur le déplacement dans tous les sens (également en oblique) d'un curseur qui laissera ou non, suivant le choix, une trace sur l'écran. Par contre, attention, vos faits et gestes sont enregistrés par l'ordinateur et ainsi il sera capable, quand vous le désirez, de refaire vos dessins de la même manière que vous les avez conçus.

La deuxième partie est l'émission-réception proprement dite. A l'émission, comme sa grande sœur en 625 lignes, balayage de l'écran ligne par ligne avec Tops de synchronisation par envoi d'une tonalité de début d'image, de fin de ligne et d'allumage de point. La fréquence de cette tonalité (environ 800 Hz) est unique, seule la durée change.

A la réception, le principe de balayage est identique et durant une ligne il y a allumage du point où se trouve le «Spot» quand une tonalité est présente. Bien sûr, pour recevoir, une interface est nécessaire et celle retenue a été la même que pour le programme CW (avec NE 567).

Avant d'entrer plus en détail dans le programme, il va falloir le rentrer au préalable dans la machine, ce qui n'est pas une mince affaire.

La première chose à faire est de créer un REM en ligne 1 d'une longueur de 1 500 octets. La meilleure solution pour effectuer ce travail est de se servir du petit programme décrit dans les pages précédentes.

Il va maintenant falloir entrer, dans ce REM, le langage machine, grâce au petit programme suivant :

```
20 REM
30 LET J=0
40 FOR I=16516 TO 16860
50 IF J>=21 THEN SCROLL
60 PRINT I,
100 INPUT K
150 PRINT K
180 POKE I,K
300 LET J=J+1
450 NEXT I
```

Il suffit de faire RUN et de taper une à une les valeurs données dans la table ci-dessous (Vidage décimal du REM). Ensuite, le programme basic pourra être entré directement (voir listing Figure 1).

Vidage décimal du REM

adresses		données			
16516	60	4	011	055	50
16521	100	04	71	100	004
16526	200	70	15	000	000
16531	100	04	71	100	004
16536	2010	254	31	31	000
16541	200	404	12	04	000
16546	17	010	0	000	001
16551	04	100	054	110	000
16556	10	000	70	10	000
16561	1000	170	40	7	000
16566	270	000	001	04	004
16571	000	001	054	100	004
16576	210	04	100	000	040
16581	00	07	04	0000	040
16586	00	101	04	0000	10
16591	04	00	045	000	101
16596	04	04	14	040	00
16601	101	04	04	00	040
16606	00	101	04	000	7
16611	04	00	000	100	111
16616	00	100	04	71	10
16621	054	000	70	10	00
16626	100	04	71	10	004
16631	40	00	000	100	000
16636	041	001	000	100	111
16641	00	4	011	000	00
16646	100	04	71	10	004
16651	000	70	15	00	100
16656	04	71	10	004	40
16661	00	004	100	000	041
16666	001	0	0	0	0
16671	0	0	0	40	10
16676	04	00	1	010	0
16681	010	054	000	100	00
16686	000	010	004	000	100
16691	40	000	000	00	00
16696	010	054	000	100	004
16701	01	00	00	11	100
16706	054	110	00	000	100
16711	177	40	10	00	11
16716	000	00	00	04	000
16721	04	100	001	001	17
16726	14	0	07	100	170
16731	00	001	001	10	000
16736	07	0	0	40	100
16741	04	00	00	004	000

16745-	00	00	00	00	00
16751-	00	00	110	00	00
16756-	00	00	00	00	00
16761-	00	00	00	00	00
16766-	17	04	00	00	00
16771-	00	110	00	00	00
16776-	00	00	17	00	00
16781-	00	00	00	110	00
16786-	07	00	00	00	00
16791-	34	00	00	00	00
16796-	00	00	00	00	00
16801-	17	00	00	00	00
16806-	110	00	00	00	00
16811-	210	17	00	00	00
16816-	00	00	110	00	00
16821-	00	00	00	17	00
16826-	00	00	00	00	00
16831-	00	00	00	00	00
16836-	00	00	00	00	00
16841-	110	00	00	00	00
16846-	00	00	00	00	00
16851-	04	140	10	00	00
16856-	00	00	00	00	00

Fig. 1 : Programme basic
TVBV

```

2 REM
3 LET P=16860
4 LET Q=P
5 CLS
6 LET E=0
10 LET A=PEEK 16396+256*(PEEK
16397)+1
15 LET F=A
18 LET D=0
30 IF PEEK A=126 THEN GOTO 200
45 LET C=27
50 POKE A,C
55 LET B=A
59 LET K=CODE INKEY$
60 LET A=A+(K=36)-(K=33)-33*(K
=35 AND A>F+32)-34*(K=30 AND A>F
+33)+33*(K=34 AND A<(F+693))+34*
(K=27 AND A<(F+693))
61 LET A=A+32*(K=63 AND A<(F+7
25))-32*(K=37 AND A>(F+33))
62 IF PEEK A=116 THEN LET A=B
63 IF K=55 THEN GOTO 2000
65 IF A<>B THEN POKE P,K
66 IF A<>B THEN LET P=P+1
67 IF PEEK A=116 THEN LET A=B
68 IF P=16012 THEN GOTO 5000
70 POKE B,D
80 IF K=29 THEN LET E=1
90 IF K=28 THEN LET E=0
95 POKE B,128*E
100 IF K=42 THEN GOTO 300
102 IF K=40 THEN GOTO 150
103 IF K=50 THEN GOTO 400
104 IF K=28 OR K=29 OR K=42 OR

```

P = adresse de rangement
des mouvements
Q = adresse de début
A = position du curseur
clignotant

Déplacement curseur



Test bordure d'écran

Test allumage ou extinction
du point considéré

```

K=40 THEN GOTO 106
105 GOTO 108
106 POKE P,K
107 LET P=P+1
108 LET K=0
109 IF P=18012 THEN GOTO 5000 P = 18012 (fin du REM)
110 GOTO 30
150 POKE P,40
155 LET P=P+1
158 IF P=18012 THEN GOTO 5000 P = 18012 (fin du REM)
160 GOTO 5
200 LET C=155
210 LET D=128*E
220 GOTO 50
300 FAST
305 POKE P,42
306 LET P=P+1
307 IF P=18012 THEN GOTO 5000 Émission de l'image présente à l'écran
308 FAST
310 RAND USR 16516
320 SLOW
330 GOTO 30
401 POKE P,255
403 LET Q=16860
405 CLS Rappel de la mémoire d'écran
407 LET W=INT (Q/256)
410 POKE 16736,W
420 POKE 16735,Q-(W*256)
430 LET Q=USR 16734
435 IF INKEY$="E" THEN GOSUB 46
0
437 IF INKEY$="S" THEN GOTO 5
440 IF INKEY$<>"C" THEN GOTO 43
5
450 IF INKEY$="" THEN GOTO 405
455 GOTO 450
460 FAST
465 RAND USR 16516
470 SLOW
480 RETURN
500 GOTO 18
600 CLS
610 LET A=1+PEEK 16396+256*PEEK 16397
620 RETURN
1000 SAVE "TVBV" (Le V de TVBV doit être tapé en vidéo normale)
1010 GOTO 5
2000 CLS
2010 RAND USR 16674
2020 IF INKEY$="C" THEN GOTO 200
0
2025 IF INKEY$="E" THEN GOTO 5
2030 GOTO 2020
5000 POKE P,255 (255) = fin de mémoire d'écran
5010 STOP

```

Quand tout le programme sera en mémoire, sauvegardez-le en faisant GOTO 1000. (Rappel : le dernier V du mot «TVBV» à la ligne 1000 doit être tapé normalement et non inversé). A la fin de la sauvegarde, le programme s'autolance et on obtient en haut à gauche de l'écran un petit point clignotant. C'est le curseur. Celui-ci symbolise la position du crayon qui, au départ, est levé. Pour le baisser, appuyez sur la touche «1» et pour le relever, sur la touche «0». Attention, l'état du crayon est mémorisé, c'est-à-dire que une fois baissé, il le restera jusqu'à l'appui sur la touche «0» pour le relever et ce même devant les déplacements.

Les déplacements se font par les touches habituelles (5, 6, 7, 8) pour des tracés horizontaux et verticaux. Par contre, 4 autres touches permettent des déplacements obliques. Ce sont :

Touches : 2 – oblique haut gauche
 Z – oblique bas gauche
 . – oblique bas droite
 9 – oblique haut droite

A l'aide de ces touches, il vous est possible donc de faire un dessin à l'écran qui est automatiquement mémorisé par l'ordinateur, non pas en considérant le fichier d'affichage, mais tous les mouvements effectués par le curseur (même les corrections).

Vous trouverez ci-après un exemple de dessin qui n'est autre que les indicatifs des auteurs.



Quand vous avez terminé un dessin, plusieurs options sont disponibles :

- **Faire un autre dessin** : Appuyez sur «C», l'écran s'efface et le curseur réapparaît en haut à gauche. Le ou les dessins précédents sont définitivement mémorisés (dans la limite de l'espace réservé dans le REM. Dans le cas présent, 1153 octets sont disponibles, ce qui permet en moyenne 3 images, ceci dépendant du nombre de mouvements nécessaires pour chacune d'elles. La limite est la longueur du REM en ligne No 1. Si vous prévoyez un REM plus grand, ce qui est tout à fait faisable, grâce au programme de création de REM, il vous faudra modifier les lignes 68, 109, 158, 307 du programme basic avec la nouvelle adresse de fin du REM. Cette adresse est donnée par : $ad\ fin = 16512 + \text{longueur du REM}$ pour un bon fonctionnement du programme.).
- **Emettre cette image** : Appuyez sur «E». Une tonalité de synchronisation sera envoyée jusqu'à l'appui sur la touche «.» qui déclenche l'émission. Ces tonalités sortent par la prise «MIC» du ZX 81. L'émission terminée, l'image est de nouveau affichée et les deux commandes précédentes de nouveau disponibles.

- **Affichage des images mémorisées** : Appuyez sur «M» et la première image qui a été mémorisée apparaît sur l'écran avec une vitesse d'exécution évidemment plus rapide que lorsque vous l'avez réalisée. A ce moment, le 3 commandes suivantes sont disponibles :
 - «C» pour visualiser l'image suivante
 - «E» pour émettre
 - «S» pour revenir à la création d'une image
- **Recevoir des images** : Appuyez sur «R» et le ZX 81 est capable de recevoir des images élaborées par un autre ZX via l'interface du décodeur morse. Lorsque la réception de l'image en cours est terminée, deux possibilités :
 - «C» pour recevoir une autre image
 - «E» pour revenir au début du programme, c'est-à-dire à la création d'une image.

L'émission ou la réception d'une image dure 20 secondes.

Vous trouverez ci-après les routines principales désassemblées de ce programme.

De 16516 à 16666 : Routine d'émission

De 16674 à 16733 : Routine de réception

De 16734 à 16856 : Routine affichage dessins mémorisés.

ÉMISSION

```

16516-LD A,N          N=4
16518-OUT N,A        N=255
16520-LD A,(NN)     NN=16514
16523-LD B,A
16524-DJNZ D        D=254 (16524)
16526-CALL NN      NN=3910
16529-RET NC
16530-LD A,(NN)    NN=16514
16533-LD B,A
16534-DJNZ D        D=254 (16534)
16536-IN A,N        N=254
16538-RRR
16539-RRR
16540-JRC D         D=230 (16516)
16542-LD HL,(NN)   NN=16395
16545-INC HL
16546-LD DE,NN     NN=725
16549-CALL NN      NN=16565
16550-LD A,(HL)
16553-CP N         N=118
16555-JRNZ D       D=16 (16573)
16557-CALL NN      NN=3910
16560-RET NC
16561-LD A,D
16562-OR E
16563-JRZ D        D=7 (16572)
16565-INC HL
16566-DEC DE
16567-CALL NN      NN=16605
16570-JR D         D=233 (16549)
16572-RET

```

Génération tonalité de synchro
avec attente appui touche «●»

HL contient l'adresse de début
du fichier d'affichage.
DE = nombre de points à
transmettre
Test fin de ligne

Test touche break

Retour si fin image (BASIC)

```

16573-CP N N=128
16575-CALL Z,NN NN=16599
16578-CALL NZ,NN NN=16599
16581-INC HL
16582-DEC DE
16583-JR D D=229 (16552)
16585-PUSH AF
16586-LD A,(NN) NN=16515
16589-RRC A
16591-JR D D=20 (16613)
16593-PUSH AF
16594-LD A,(NN) NN=16515
16597-JR D D=14 (16613)
16599-PUSH AF
16600-LD A,(NN) NN=16515
16603-JR D D=33 (16638)
16605-PUSH AF
16606-LD A,(NN) NN=16515
16609-RLC A
16611-JR D D=25 (16638)
16613-PUSH HL
16614-PUSH BC
16615-LD L,A
16616-LD A,(NN) NN=16514
16619-LD B,A
16620-DJNZ D D=254 (16620)
16622-CALL NN NN=9910
16625-LD A,(NN) NN=16514
16628-LD B,A
16629-DJNZ D D=254 (16629)
16631-DEC L
16632-JRNZ D D=236 (16616)
16634-POP BC
16635-POP HL
16636-POP AF
16637-RET
16638-PUSH HL
16639-PUSH BC
16640-LD L,A
16641-LD A,N N=4
16643-OUT N,A N=255
16645-LD A,(NN) NN=16514
16648-LD B,A
16649-DJNZ D D=254 (16649)
16651-CALL NN NN=9910
16654-LD A,(NN) NN=16514
16657-LD B,A
16658-DJNZ D D=254 (16658)
16660-DEC L
16661-JRNZ D D=234 (16641)
16663-POP BC
16664-POP HL
16665-POP AF
16666-RET
16667-NOP
16668-NOP
16669-NOP

```

Lecture fichier d'affichage :
tonalité si 128, sinon silence

Start ligne = 1/2 bit silence

Synchro ligne = 2 bits tonalité
1 bit \cong 28 ms

Génération d'un silence

Génération d'une tonalité

```

16670-NOP
16671-NOP
16672-NOP
16673-NOP
16674-LD HL, (NN)      NN=16695
16677-INC HL
16678-LD BC, NN        NN=1725
16681-IN A, N          N=254
16683-AND N            N=128
16685-JR NZ D          D=250 (16681)
16687-IN A, N          N=254
16689-AND N            N=128
16691-JR Z D           D=250 (16687)
16693-CALL NN         NN=16725
16696-IN A, N          N=254
16698-AND N            N=128
16700-CALL Z, NN      NN=16721
16703-INC HL
16704-DEC BC
16705-LD A, (HL)
16706-CP N             N=115
16708-JR NZ D          D=239 (16693)
16710-LD A, B
16711-OR C
16712-JR Z D           D=10 (16724)
16714-INC HL
16715-DEC BC
16716-CALL NN         NN=16725
16719-JR D             D=238 (16687)
16721-LD (HL), N      N=128
16723-RET
16724-RET
16725-LD DE, NN       NN=1600
16728-DEC DE
16729-LD A, D
16730-OR E
16731-JR NZ D         D=251 (16728)
16733-RET

```

RÉCEPTION

HL = adresse début affichage
BC = nombre de points

Attente fin tonalité synchro
image ou ligne

Appel tempo 1 bit

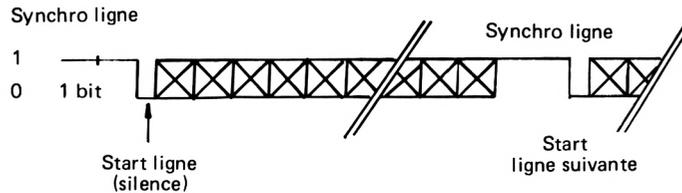
Allumage point si tonalité
+test fin de ligne

Si fin image, retour au basic

Tempo

Format d'une ligne

1 = Tonalité, 0 = Silence



MÉMOIRE

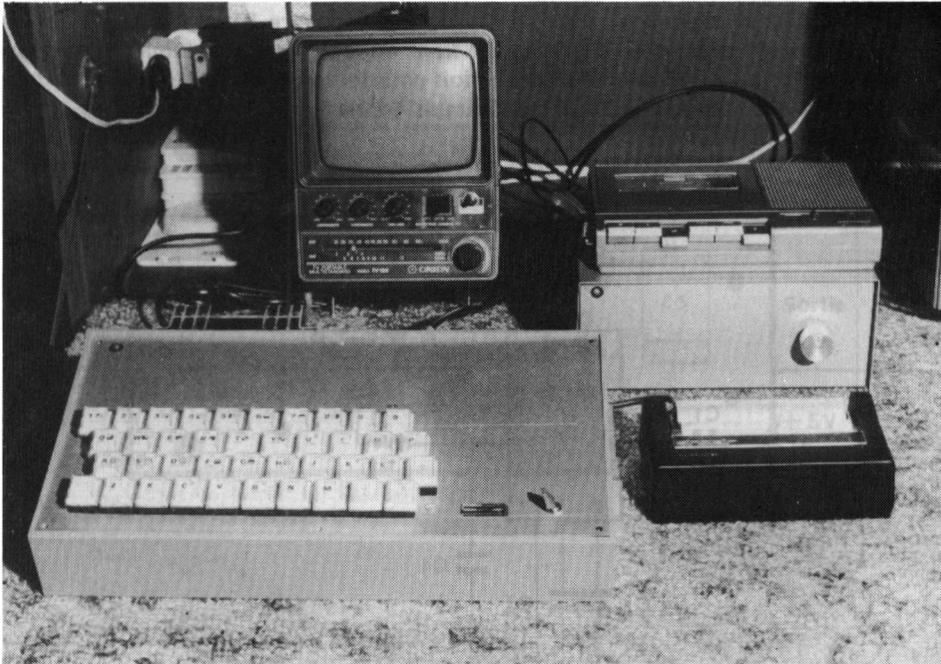
16734-LD DE,NN	NN=17387
16737-LD B,N	N=0
16739-LD HL,(NN)	NN=16396
16742-INC HL	
16743-LD A,(DE)	
16744-CP N	N=255
16746-RET Z	
16747-CP N	N=29
16749-JRNZ D	D=2 (16753)
16751-LD B,N	N=128
16753-LD (HL),B	
16754-CP N	N=28
16756-JRNZ D	D=3 (16761)
16758-LD B,N	N=0
16760-LD (HL),B	
16761-CP N	N=30
16763-JRNZ D	D=8 (16773)
16765-PUSH DE	
16766-LD DE,NN	NN=34
16769-SBC HL,DE	
16771-POP DE	
16772-LD (HL),B	
16773-CP N	N=37
16775-JRNZ D	D=8 (16785)
16777-PUSH DE	
16778-LD DE,NN	NN=32
16781-SBC HL,DE	
16783-POP DE	
16784-LD (HL),B	
16785-CP N	N=27
16787-JRNZ D	D=7 (16796)
16789-PUSH DE	
16790-LD DE,NN	NN=34
16793-ADD HL,DE	
16794-POP DE	
16795-LD (HL),B	
16796-CP N	N=63
16798-JRNZ D	D=7 (16807)
16800-PUSH DE	
16801-LD DE,NN	NN=32
16804-ADD HL,DE	
16805-POP DE	
16806-LD (HL),B	
16807-CP N	N=35
16809-JRNZ D	D=8 (16819)
16811-PUSH DE	
16812-LD DE,NN	NN=33
16815-SBC HL,DE	
16817-POP DE	
16818-LD (HL),B	
16819-CP N	N=34
16821-JRNZ D	D=7 (16830)
16823-PUSH DE	
16824-LD DE,NN	NN=33
16827-ADD HL,DE	
16828-POP DE	
16829-LD (HL),B	

DE = adresse début image mémorisée
HL = adresse début affichage

255 = fin REM ⇒ retour basic

}		Test allumage point (1)
}		Test extinction point (0)
}		Test déplacement haut gauche (2)
}		Test déplacement haut droite (9)
}		Test déplacement bas droite (●)
}		Test déplacement bas gauche (Z)
}		Test déplacement haut (7)
}		Test déplacement bas (6)

16830-CP N	N=33	} Test déplacement gauche (5)
16832-JRNZ D	D=2 (16836)	
16834-DEC HL		
16835-LD (HL),B		} Test déplacement droite (8)
16836-CP N	N=36	
16838-JRNZ D	D=2 (16842)	
16840-INC HL		} Si autre valeur, retour basic
16841-LD (HL),B		
16842-CP N	N=40	
16844-JRZ D	D=7 (16853)	} Sinon continue
16846-CP N	N=42	
16848-JRZ D	D=3 (16853)	
16850-INC DE		} Le retour au basic se fait avec l'adresse d'arrêt dans BC.
16851-JR D	D=146 (16743)	
16853-INC DE		
16854-PUSH DE		
16855-POP BC		
16856-RET		



«ERCW» émission et réception MORSE

Voici un programme écrit entièrement en langage machine qui permettra de voir quelques unes des manières possibles pour aborder un problème donné. Même les commandes d'impression de messages ont été préprogrammées en langage machine.

Ce programme permet d'émettre et de recevoir en télégraphie MORSE à partir du ZX 81, moyennant la réalisation d'un circuit interface disposé entre le ZX et le récepteur de trafic.

Il faudra néanmoins effectuer une modification (mineure) sur le ZX 81, qui consiste à souder un fil au point commun C10 (10nF) et R33 (4,7 k Ω) à l'entrée LOAD (EAR) du ZX 81.

Ce fil sera blindé (tresse soudée à la masse) et terminé par une prise reliée à la sortie de l'interface.

Nous vous proposons un schéma possible du montage interface (voir figure 15). Il peut se connecter sur le potentiomètre BF du récepteur ou sur une sortie casque ou H.P. Dans ce cas, il est conseillé de disposer entre le point E et la masse 2 diodes, montées tête-bêche, en protection d'un fort niveau accidentiel.

Les réglages sont simples, voire inexistants. Le potentiomètre P1 sert à régler la sensibilité du montage et sera ajusté pour un clignotement franc de la diode électroluminescente au rythme de la télégraphie. Pour P2, qui sera pré-réglé une fois pour toutes, et qui agit sur la plage de capture de la PLL, il sera préférable de choisir un modèle multi-tours.

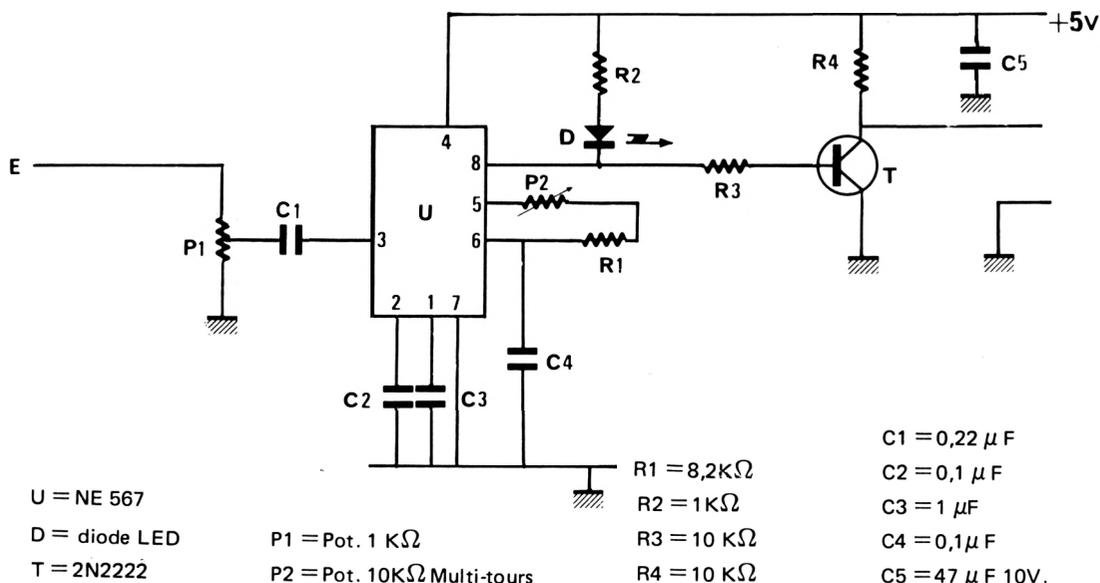


FIGURE 15 : INTERFACE DECODEUR MORSE

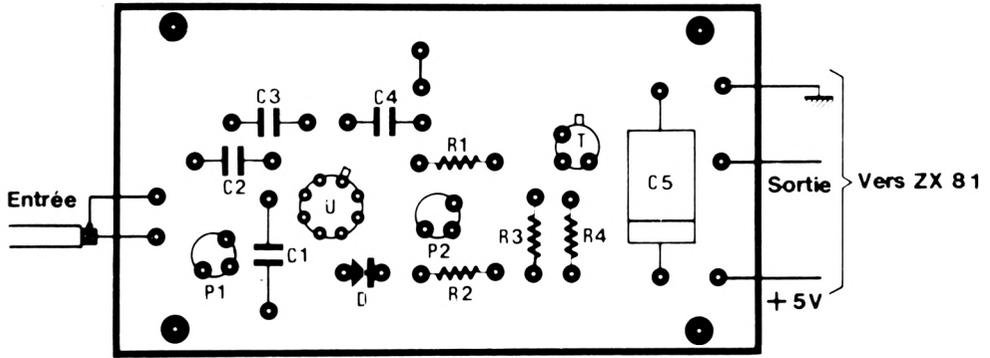
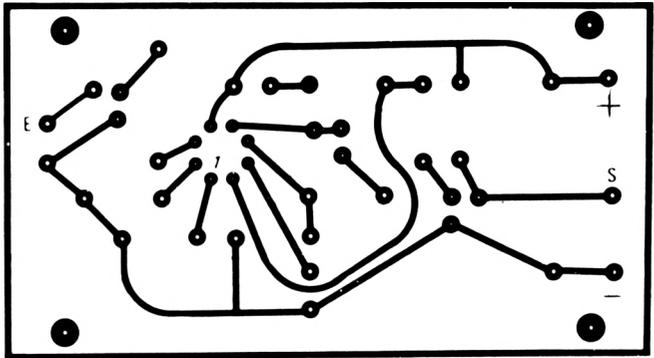


FIGURE 15 : INTERFACE DECODEUR MORSE.

LE MESSAGE EN MEMOIRE EST :

PROGRAMME DE DEMONSTRATION DU PA
R MICRO ORDINATEUR ZX 81. CONCU P
R FIEZH EDDY ET F6GKQ DENIS, IL
PERMET DE TRANSMETTRE ET DE RECE
VOIR EN TELEGRAPHIE MORSE. IL POS
SEDE UNE FONCTION DICTEE POUR LA
PREPARATION A LA LICENCE COMPLE
TE. BON TRAFIC AVEC CODECW 30 +K

E POUR ECRIRE UN MESSAGE
T POUR EMETTRE MESSAGE MEMORISE
M POUR VOUS ENTRAINER
S POUR ARRETER
R POUR RECEVOIR

Présentation du programme dans le mode Emission.

Ci dessous, la règle à respecter : pas de touche autre que les touches alphanumériques et de ponctuation. En cas d'erreur, pas de tentative de retour arrière ou de RUBOUT.

Pour terminer le message faire > (shift M).

LE MESSAGE EN MEMOIRE EST :

POUR ECRIRE UN MESSAGE IL NE
FAUT UTILISER QUE LES TOUCHES
ALPHANUMERIQUES ET DE PONCTUATIO
N. UNE ERREUR NE DOIT PAS ETRE
CORRIGEE PAR UN RETOUR EN ARRIE
RE DU CURSEUR EN UTILISANT LES
FLECHES. L'ASTROPHIE EST SUR LA
TOUCHE P SHIFTEE. POUR CLORE UN
MESSAGE APPUYER SUR SHIFT M

E POUR ECRIRE UN MESSAGE
T POUR EMETTRE MESSAGE MEMORISE
M POUR VOUS ENTRAINER
S POUR ARRETER
R POUR RECEVOIR

Après une dictée d'entraînement, le corrigé apparaît en faisant C, puis E dans le message en mémoire.

Examen du Programme «ERCW»

Ce programme est très long puisqu'il occupe environ 2 800 octets. Vous créez donc un REM, automatiquement, au moyen du programme de création de REM livré précédemment, et vous ferez suivre ce REM d'un REM vide, puis de l'instruction suivante :

RAND USR 17056

A partir de là, écrivez une boucle d'entrée des instructions dont la liste vous est donnée en hexadécimal (inspirez-vous du petit programme donné en exemple dans le chapitre d'écriture en langage machine, et soyez vigilants lors de l'introduction des instructions). Sauvegardez par petites parties et surtout, ne tentez pas d'essayer le programme avant une sauvegarde complète.

Voici comment se décompose ce programme «ERCW»

16519	à	16742	Routine de réception
16750	à	16871	Table de transcodage réception
16873	à	16889	Routine position écriture
16900	à	16960	Routine de saisie de caractères
16970	à	17033	Table de codage émission
17040	à	17050	Routine d'impression du message mémorisé
17056	à	17186	Point d'entrée du programme ; présentation ; aiguillage
17190	à	17208	Lecture du clavier
17230	à	17244	Routine d'affichage des textes sur écran
17256	à	17374	Routine d'émission
17381	à	17733	Zone de rangement du message transcodé à émettre
17796	à	18147	Zone de rangement du message, en clair, à émettre
18169	à	18722	Stockage des «PRINT» de présentation
18728	à	18827	Présentation Réception
18834	à	19002	Présentation Emission
19011	à	19020	Table des vitesses émission
19021	à	19069	Suite routine émission
19071	à	19139	Routine de modification vitesse de décodage
19189	à	19308	Création de la dictée aléatoire

Avant de sauvegarder le programme et de le lancer pour le première fois, il faudra faire :

```
POKE 17382,255
POKE 17797,18
POKE 17798,255
```

Le programme, après RUN, demande le choix émission-réception. En réception il faut lui indiquer approximativement la vitesse (de 9 à 0 pour très lent à très rapide). Les vitesses les plus habituelles se situent entre 6 et 2. La vitesse peut être modifiée en cours de décodage, à l'aide des touches et , en «temps réel».

Un appui sur la touche . (point) provoque la sortie du programme de décodage.

A l'émission, les vitesses sont «pré-réglées» par des valeurs situées dans une table (19011 à 19020). On choisira la vitesse de 0 à 9.

Voici l'ordre de grandeur de vitesse, en mots/minute, en fonction de la valeur rangée dans la table de 19011 à 19020.

Adresse	Valeur	Vitesse (mots/mn)
19011	140	6
19012	89	10
19013	64	15
19014	49	20
19015	38	25
19016	32	30
19017	24	40
19018	19	50
19019	12	75
19020	10	100

Vous pourrez modifier ces valeurs selon vos besoins.

Fonctionnement du programme

Nous avons utilisé quelques routines de la ROM et certaines possibilités du ZX.

- Lecture de la prise «EAR» (LOAD (en 16554)
IN A, 254
RLA
JRNC

On lit par une instruction d'entrée, on effectue une rotation par la gauche et on teste le bit de retenue.

- Sortie sur la prise «MIC» (SAVE (en 17297)
OUT 255, A

Permet de générer une tonalité (en répétant cette opération dans une boucle) sur la prise MIC (opération qui doit être suivie d'un IN A, 254 dans le ZX).

- Affichage par RST 16 (RST 10H) (en 17047)

Dans le ZX, il est possible d'afficher simplement un caractère (codes 0 à 63) en le mettant dans l'accumulateur et en faisant RST 16. On peut «planter» le système si on tente d'afficher un caractère interdit.

- Affichage par écriture dans le fichier d'affichage (de 17230 à 17242)

On doit d'abord s'assurer que le pointeur d'écriture (ici c'est HL) n'est pas sur un caractère de fin de ligne (code 118). Si c'est le cas, on incrémente d'une position.

Cette précaution prise, on charge le code du caractère à afficher (il est ici pointé par DE et transféré dans A) à la position indiquée par HL. Avec ce procédé, on peut même aller écrire sur les deux lignes interdites du bas de l'écran (voyez l'exemple du programme de RTTY où on affiche au bas de l'écran les options en vidéo inversée).

– Routine d'exploration du clavier (en 17195)

On fait appel à la routine située en ROM à l'adresse 699. La «valeur» de la touche est contenue dans HL. Pour obtenir le code «SINCLAIR» de la touche, on appelle la routine ROM située en 1981, après avoir transféré la valeur de HL à BC. Le code est disponible alors dans l'accumulateur si on fait :

LD A, (HL) (ex : en 17207)

– Routine d'effacement écran (en 18728)

Là, c'est très simple. On fait appel à la routine CLS, localisée à l'adresse 2602 de la ROM. On aurait pu aussi écrire une routine complète comme cela a été fait dans le programme RTTY.

Pour imprimer les textes à l'écran (menu, présentation, options, etc...) on a rangé les chaînes de caractères à un endroit précis de la mémoire (18169 à 18722) et on vient les prélever. Le principe est très simple. Prenons l'exemple de 18761 à 18773.

On charge le registre B avec le nombre de caractères à transférer sur l'écran (LD B, 27).

On pointe le début de fichier d'affichage (son adresse est donnée par D-FILE en 16396) d'où le LD HL, (16396).

On additionne un «offset» correspondant au nombre de lignes à sauter sur l'écran (ou des positions dans le fichier d'affichage si on ne veut pas écrire au début d'une ligne). Ici on a LD DE, 132 (132 = 4 x 33) et ADD HL, DE qui positionne le pointeur d'écriture à D-FILE + 132.

On pointe avec DE l'adresse du premier caractère de la chaîne à imprimer (LD DE, 18327).

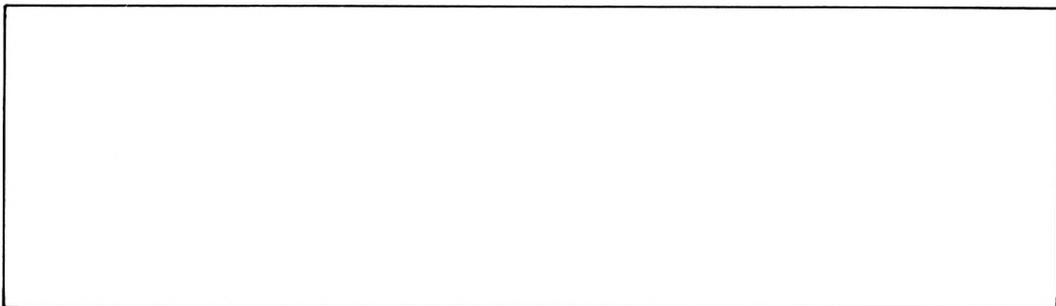
On appelle à la routine d'écriture en 17230.

Cette routine écrit directement le caractère pointé par DE à l'endroit pointé par HL (si cet emplacement ne contient pas 118).

Nous ne décrivons pas toutes les autres routines utilisées par ce programme et vous invitons à consulter les listings en ASSEMBLEUR pour prendre des exemples d'utilisation des instructions.

D'autres caractéristiques de la ROM ZX ont été utilisées comme en 18902 l'appel à la routine STOP (3292), en 18795 l'appel de la routine FAST (3875), en 19051 le passage en mode SLOW (3883).

Tous ces exemples peuvent être reconsidérés pour l'utilisation dans d'autres programmes.



PROGRAMME ERCW

```

16519-NOP
16520-NOP
16521-NOP
16522-NOP
16523-LD A,N
16524-LD (NN),A N=128
16525-LD A,N NN=16515
16526-LD A,N N=0
16527-LD (NN),A NN=16516
16528-LD A,N N=21
16529-LD (NN),A NN=16517
16530-LD (NN),A NN=16518
16541-LD A,N N=0
16543-LD (NN),A NN=16514
16546-LD DE,NN NN=0
16549-IN A,N N=254
16551-RLA
16552-JRNC D D=251 (16549)
16554-IN A,N N=254
16556-RLA
16557-JRNC D D=251 (16554)
16559-LD A,N N=2
16561-LD B,N N=75
16563-DJNZ D D=254 (16563)
16565-DEC A
16566-JRNZ D D=249 (16561)
16568-LD A,N N=127
16570-IN A,N N=254
16572-RRR
16573-RRR
16574-RET NC
16575-NOP
16576-NOP
16577-LD A,(NN) NN=16515
16580-LD B,A
16581-IN A,N N=254
16583-AND N N=128
16585-CP B
16586-JRNZ D D=15 (16583)
16588-LD A,(NN) NN=16516
16591-CP N N=255
16593-JRZ D D=15 (16510)
16595-INC A
16596-LD (NN),A NN=16516
16599-JR D D=214 (16559)
16601-NOP
16602-NOP
16603-LD (NN),A NN=16515
16606-AND N N=128
16608-JRNZ D D=23 (16633)
16610-LD A,(NN) NN=16518
16613-RLC A
16615-LD B,A
16616-LD A,(NN) NN=16516
16619-CP B
16620-JRNC D D=75 (16697)
16622-RLC D
16624-RLC E

```

PARTIE RÉCEPTION

Initialisation des variables
16514 1 = trait 0 = point
16515 1 = élément
0 = espace
16516 durée de l'élément
16517 durée de l'élément
précédent
16518 durée du point

Attente d'un espace

Attente d'un élément

Tempo de référence temps

Teste l'appui sur «●» pour
arrêt du décodage

Lecture du port d'entrée
Compare à la lecture pré-
cédente. Si résultat différent
mesure la durée du signal
présent

Sauvergarde le type du signal
(élément ou espace)
Si c'est un espace, détermine
si espace inter-éléments ou
inter-caractères

16626-LD A,N	N=0
16628-LD (NN),A	NN=16516
16631-JR D	D=216 (16595)
16633-LD A,(NN)	NN=16516
16636-LD C,A	
16637-RLC A	
16639-ADD A,C	
16640-LD B,A	
16641-LD A,(NN)	NN=16517
16644-RLC A	
16646-LD (NN),A	NN=16517
16649-CP B	
16650-JRNC D	D=31 (16683)
16652-LD A,(NN)	NN=16516
16655-LD B,A	
16656-LD A,(NN)	NN=16517
16659-CP B	
16660-JRC D	D=7 (16669)
16662-LD A,(NN)	NN=16514
16665-AND N	N=255
16667-JRZ D	D=14 (16683)
16669-LD A,N	N=1
16671-LD (NN),A	NN=16514
16674-INC E	
16675-LD A,(NN)	NN=16516
16676-LD (NN),A	NN=16517
16681-JR D	D=199 (16626)
16683-LD A,(NN)	NN=16516
16686-LD (NN),A	NN=16518
16689-LD A,N	N=0
16691-LD (NN),A	NN=16514
16694-INC D	
16695-JR D	D=234 (16675)
16697-LD A,D	
16698-RLC A	
16700-ADC A,E	
16701-ADD A,N	N=110
16703-LD H,N	N=65
16705-LD L,A	
16706-LD B,(HL)	
16707-LD A,B	
16708-CP N	N=64
16710-JRNC D	D=24 (16736)
16712-RST 16	
16713-LD DE,NN	NN=0
16716-LD A,(NN)	NN=16518
16719-RLC A	
16721-RLC A	
16723-ADD A,A	
16724-NOP	
16725-NOP	
16726-NOP	
16727-LD B,A	
16728-LD A,(NN)	NN=16516
16731-CP B	
16732-JRNC D	D=2 (16736)
16734-JR D	D=3 (16739)
16736-LD A,N	N=0
16738-RST 16	
16739-CALL NN	NN=16873
16742-JR D	D=138 (16626)

Espace inter-éléments : préparation pour prochaine lecture.

Cas d'un élément :
détermine le type : trait -
point

Effectue la sauvegarde dans
les registres respectifs.
Si c'est un point, sauvegarde
sa durée jusqu'à la mesure
d'un prochain point.

Routine d'affichage

Recherche dans la table, le code
SINCLAIR du caractère recon-
nu et teste sa validité

Affiche le caractère

Espace entre mots

```

100073-AND A
100074-LD A, (NN)
100077-CP N
100079-JRNC D
100081-PUSH DE
100080-CALL NN
100085-POP DE
100086-CALL NN
100089-RET
100090-NOP
100091-NOP
100092-NOP
100093-NOP
100094-NOP
100095-NOP
100096-NOP
100097-NOP
100098-NOP
100099-NOP
100000-NOP
100001-LD DE, NN
100004-PUSH DE
100005-LD BC, NN
100008-PUSH BC
100009-CALL NN
100100-LD B, H
100100-LD C, L
100104-LD A, C
100105-INC A
100106-JRZ D
100108-CALL NN
100201-LD A, (HL)
100202-RST 16
100203-POP BC
100204-LD (BC), A
100205-INC BC
100206-PUSH BC
100207-LD B, A
100208-CP N
100300-JRZ D
100302-LD A, B
100303-ADD A, N
100309-LD H, N
100307-LD L, A
100308-LD A, (HL)
100309-POP BC
100400-POP DE
100401-LD (DE), A
100402-INC DE
100403-LD A, N
100405-LD (DE), A
100406-PUSH DE
100407-PUSH BC
100408-CALL NN
100501-LD A, L
100502-INC A
100503-JRNZ D
100505-JR D
100507-POP BC
100508-POP DE
100509-RET

```

```

NN=16442
N=4
D=5 (16886)

```

```

NN=2602
NN=19071

```

```

NN=17381

```

```

NN=17796

```

```

NN=699

```

```

D=247 (16909)
NN=1981

```

```

N=18
D=25 (16957)

```

```

N=74
N=66

```

```

N=255

```

```

NN=699

```

```

D=249 (16948)
D=208 (16909)

```

Teste si on est rendu à la dernière ligne écran et efface l'écran dans ce cas pour continuer le décodage.
Appelle routine «modification vitesse»

Introduction du message

DE pointe le début message transcodé
BC pointe le début du message à transcoder

Lit le clavier tant qu'aucune touche n'est pressée, sinon

appelle routine décodage clavier, affiche, sauvegarde.

Teste si fin de message
(18 = >)

Effectue la transformation du code SINCLAIR en un code exploitable par le sous-programme émission pour la génération des points-traités.

Anti-rebond du clavier

17040-LD HL, NN	NN=17796
17043-LD A, (HL)	
17044-CP N	N=18
17046-RET Z	
17047-RST 16	
17048-INC HL	
17049-JR D	D=248 (17043)
17051-NOP	
17052-NOP	
17053-NOP	
17054-NOP	
17055-NOP	
17056-LD B, N	N=14
17058-LD HL, (NN)	NN=16396
17061-LD DE, NN	NN=275
17064-ADD HL, DE	
17065-LD DE, NN	NN=18169
17068-CALL NN	NN=17230
17071-LD B, N	N=14
17073-LD HL, (NN)	NN=16396
17076-LD DE, NN	NN=473
17079-ADD HL, DE	
17080-LD DE, NN	NN=18169
17083-CALL NN	NN=17230
17086-LD B, N	N=14
17088-LD HL, (NN)	NN=16396
17091-LD DE, NN	NN=308
17094-ADD HL, DE	
17095-LD DE, NN	NN=18196
17098-CALL NN	NN=17230
17101-LD B, N	N=14
17103-LD HL, (NN)	NN=16396
17106-LD DE, NN	NN=341
17109-ADD HL, DE	
17110-LD DE, NN	NN=18183
17113-CALL NN	NN=17230
17116-LD B, N	N=14
17118-LD HL, (NN)	NN=16396
17121-LD DE, NN	NN=374
17124-ADD HL, DE	
17125-LD DE, NN	NN=18209
17128-CALL NN	NN=17230
17131-LD B, N	N=14
17133-LD HL, (NN)	NN=16396
17136-LD DE, NN	NN=407
17139-ADD HL, DE	
17140-LD DE, NN	NN=18183
17143-CALL NN	NN=17230
17146-LD B, N	N=14
17148-LD HL, (NN)	NN=16396
17151-LD DE, NN	NN=440
17154-ADD HL, DE	
17155-LD DE, NN	NN=18223
17158-CALL NN	NN=17230
17161-LD B, N	N=48
17163-LD HL, (NN)	NN=16396
17166-LD DE, NN	NN=660
17169-ADD HL, DE	
17170-LD DE, NN	NN=18237
17173-CALL NN	NN=17230
17176-CALL NN	NN=17190
17179-CP N	N=42
17181-JPZ NN	NN=18834
17184-JP NN	NN=18728

**Sous-programme
affichage message**

Test de fin de message
Affiche
Boucle si non terminé

**Début du programme
principal**

Affiche le «pavé» de pré-
sentation et le choix Émis-
sion-Réception

17187-NOP
 17188-NOP
 17189-NOP
 17190-NOP
 17191-NOP
 17192-NOP
 17193-NOP
 17194-NOP
 17195-CALL NN
 17198-LD B,H
 17199-LD C,L
 17200-LD A,C
 17201-INC A
 17202-JRZ D
 17204-CALL NN
 17207-LD A,(HL)
 17208-RET

NN=699

Routine exploration clavier

Lit le clavier et attend l'appui sur une touche

D=242 (17190)
 NN=1981

Décodage de la touche

17230-LD A,N
 17232-CP (HL)
 17233-JRNZ D
 17235-INC HL
 17236-NOP
 17237-NOP
 17238-LD A,(DE)
 17239-LD (HL),A
 17240-INC DE
 17241-INC HL
 17242-DJNZ D
 17244-RET

N=118

D=3 (17238)

Routine d'écriture dans le fichier d'affichage, des chaînes de caractères à imprimer

D=242 (17230)

17256-LD HL,NN
 17259-LD A,(HL)
 17260-CP N
 17262-JRNZ D
 17264-INST/IY
 17264-253/54/124/5 LD(16508),5
 17268-JR D
 17270-CP N
 17272-RET Z
 17273-LD D,N
 17275-DEC D
 17276-RLA
 17277-JRNC D
 17279-DEC D
 17280-JRZ D
 17282-RLA
 17283-JRNC D
 17285-LD B,N
 17287-JR D
 17289-LD B,N
 17291-EX AF,AF"
 17292-INST/IY

NN=17381

N=0
 D=6 (17270)

D=59 (17329)

N=255

N=9

D=252 (17275)

D=43 (17325)

D=34 (17319)

N=3

D=2 (17291)

N=1

SOUS-PROGRAMME ÉMISSION

HL pointe début message transcodé

Charge nombre d'unités de temps pour un espace

Teste si fin du message (255)

D = compteur de décalages

La fin du caractère à émettre est reconnue quand D = 0

17292	-253/94/123	LDE (16507)	
17295	-LD C,N	N=127	
17297	-OUT Z,A	N=255	16507 contient la vitesse d'émission
17299	-DEC D		
17300	-JRNZ D	D=253 (17299)	
17302	-LD C,N	N=127	
17304	-IN A,N	N=254	Exécute boucle sortie de tonalité sur la prise SAVE (MIC) du ZX
17306	-DEC A		
17307	-JRNZ D	D=253 (17306)	
17309	-DEC E		
17310	-JRNZ D	D=239 (17295)	
17312	-DJNZ D	D=234 (17292)	
17314	-EX AF,AF"		
17315	-INST/IY		1 espace entre bits d'un même caractère
17315	-253/54/124/1	LD (16508),1	
17319	-CALL NN	NN=17358	
17322	-LR D	D=211 (17279)	
17324	-NOP		
17325	-INST/IY		3 espaces entre 2 caractères différents
17325	-253/54/124/3	LD (16508),3	
17329	-CALL NN	NN=17358	
17332	-INC HL		
17333	-LD A,N	N=127	
17335	-IN A,N	N=254	Test appui sur touche BREAK
17337	-ARRA		
17338	-RET NC		
17339	-LR D	D=174 (17259)	Au caractère suivant
17341	-NOP		
17342	-NOP		
17343	-NOP		
17344	-NOP		
17345	-NOP		
17346	-NOP		
17347	-NOP		
17348	-NOP		
17349	-NOP		
17350	-NOP		
17351	-NOP		
17352	-NOP		
17353	-NOP		
17354	-NOP		
17355	-NOP		
17356	-NOP		
17357	-NOP		
17358	-INST/IY		Dans E, délai en nombre d'unités
17358	-253/94/123	LDE, (16507)	
17361	-LD B,N	N=243	
17363	-NOP		
17364	-DJNZ D	D=253 (17363)	Entre bits d'un caractère : 1 unité
17366	-DEC E		Entre 2 caractères : 3 unités
17367	-JRNZ D	D=248 (17361)	Entre 2 mots : 7 unités
17369	-INST/IY		
17369	-253/53/124	DEC, (16508)	
17372	-JRNZ D	D=240 (17358)	
17374	-RET		

Partie préparation RÉCEPTION

18738-CALL NN	NN=2602	
18739-LD B,N	N=21	Efface l'écran
18739-LD HL,(NN)	NN=16396	
18739-LD DE,NN	NN=1	
18739-ADD HL,DE		Écrit les différentes instructions de préparation
18740-LD DE,NN	NN=18285	
18740-CALL NN	NN=17230	
18740-LD B,N	N=21	
18740-LD HL,(NN)	NN=16396	
18751-LD DE,NN	NN=66	
18754-ADD HL,DE		
18755-LD DE,NN	NN=18306	
18756-CALL NN	NN=17230	Connection interface
18761-LD B,N	N=27	
18763-LD HL,(NN)	NN=16396	
18769-LD DE,NN	NN=132	Choix de la vitesse
18769-ADD HL,DE		
18770-LD DE,NN	NN=18327	
18773-CALL NN	NN=17230	
18776-CALL NN	NN=17190	Appelle routine lecture clavier
18779-SUB N	N=28	
18781-CP N	N=0	
18783-JRZ D	D=5 (18790)	Vitesse rangée en 16560
18789-LD (NN),A	NN=16560	Si 0 on passera en FAST
18788-JR D	D=8 (18798)	
18790-INC A		
18791-INC A		
18792-LD (NN),A	NN=16560	
18799-CALL NN	NN=3875	Mode FAST (3875)
18799-CALL NN	NN=2602	CLS (2602)
18801-CALL NN	NN=16519	Saut du décodage (16519)
18804-LD B,N	N=24	
18806-LD HL,(NN)	NN=16396	
18809-LD DE,NN	NN=726	
18812-ADD HL,DE		
18813-LD DE,NN	NN=18354	Écrit «Presser C pour continuer»
18816-CALL NN	NN=17230	
18819-JP NN	NN=19051	Saut en «Recommence»
18822-NOP		
18823-NOP		
18824-NOP		
18825-NOP		
18826-NOP		
18827-NOP		
18834-CALL NN	NN=2602	Efface l'écran puis indique
18837-LD B,N	N=26	
18839-LD HL,(NN)	NN=16396	
18842-LD DE,NN	NN=1	
18845-ADD HL,DE		le message en mémoire
18846-LD DE,NN	NN=18493	
18849-CALL NN	NN=17230	
18852-LD B,N	N=26	
18854-LD HL,(NN)	NN=16396	
18857-LD DE,NN	NN=34	
18860-ADD HL,DE		
18861-LD DE,NN	NN=18519	
18864-CALL NN	NN=17230	
18867-LD HL,(NN)	NN=16396	
18870-LD DE,NN	NN=133	
18873-ADD HL,DE		

100674	-LD (NN),HL	NN=16398
100677	-CALL NN	NN=17240
100680	-LD B,N	N=143
100682	-LD HL,(NN)	NN=16396
100685	-LD DE,NN	NN=562
100688	-ADD HL,DE	
100690	-LD DE,NN	NN=16545
100693	-CALL NN	NN=17230
100695	-CALL NN	NN=17190
100698	-CP N	N=96
100700	-LRZ D	D=3 (18905)
100702	-CALL NN	NN=3292
100705	-CP N	N=95
100707	-LRZ NN	NN=18728
100710	-CP N	N=98
100712	-LRZ NN	NN=19189
100715	-CP N	N=97
100717	-LRZ D	D=48 (18967)
100719	-CALL NN	NN=2602
100720	-LD B,N	N=20
100724	-LD HL,(NN)	NN=16396
100727	-LD DE,NN	NN=1
100730	-ADD HL,DE	
100731	-LD DE,NN	NN=18379
100734	-CALL NN	NN=17230
100737	-LD B,N	N=13
100739	-LD HL,(NN)	NN=16396
100742	-LD DE,NN	NN=353
100745	-ADD HL,DE	
100746	-LD DE,NN	NN=18399
100749	-CALL NN	NN=17230
100752	-LD HL,(NN)	NN=16396
100755	-LD DE,NN	NN=34
100758	-ADD HL,DE	
100759	-LD (NN),HL	NN=16398
100762	-CALL NN	NN=16900
100765	-NOP	
100768	-NOP	
100771	-NOP	
100774	-NOP	
100777	-NOP	
100780	-NOP	
100783	-NOP	
100786	-NOP	
100789	-NOP	
100792	-NOP	
100795	-NOP	
100798	-NOP	
100801	-CALL NN	NN=2602
100804	-LD B,N	N=30
100807	-LD HL,(NN)	NN=16396
100810	-LD DE,NN	NN=660
100813	-ADD HL,DE	
100816	-LD DE,NN	NN=18413
100819	-CALL NN	NN=17230
100822	-CALL NN	NN=17190
100825	-ADD A,N	N=39
100828	-LD L,A	
100831	-LD H,N	N=74
100834	-LD A,(HL)	
100837	-LD (NN),A	NN=16507
100840	-JR D	D=18 (19021)
100843	-NOP	
100846	-NOP	
100849	-NOP	
100852	-NOP	
100855	-NOP	
100858	-NOP	
100861	-NOP	

Va afficher le message en mémoire puis propose les différentes options

Teste le clavier
Détermine l'option choisie

Permet l'introduction d'un nouveau message (16900)

Appelle routine exploration clavier,

charge la vitesse (table 19012)
Modifie vitesse décodage

```

19021 -NOP
19022 -NOP
19023 -NOP
19024 -NOP
19025 -NOP
19026 -NOP
19027 -NOP
19028 -NOP
19029 -NOP
19030 -NOP
19031 -NOP
19032 -NOP
19033 -NOP
19034 -NOP
19035 -NOP
19036 -CALL NN      NN=17190
19039 -CALL NN      NN=3875
19042 -CALL NN      NN=2682
19045 -CALL NN      NN=17256
19048 -LJ NN        NN=18804
19051 -CALL NN      NN=3883
19054 -CALL NN      NN=17190
19057 -CP N         N=46
19059 -JRNZ D       D=249 (19054)
19061 -NOP
19062 -NOP
19063 -NOP
19064 -CALL NN      NN=2682
19067 -LJ NN        NN=17056
19070 -RET

19071 -PUSH DE
19072 -CALL NN      NN=699
19075 -LD B,H
19076 -DEC L
19077 -LD A,C
19078 -INC A
19079 -JNZ D        D=47 (19128)
19081 -CALL NN      NN=1981
19084 -LD A,(HL)
19085 -CP N         N=35
19087 -JRNZ D       D=9 (19098)
19089 -LD A,(NN)   NN=16560
19092 -INC A
19093 -NOP
19094 -NOP
19095 -NOP
19096 -JR D         D=20 (19118)
19098 -CP N         N=34
19100 -JRNZ D       D=19 (19121)
19102 -LD A,(NN)   NN=16560
19105 -DEC A
19106 -JRNZ D       D=10 (19118)
19108 -INC A
19109 -INC A
19110 -LD (NN),A   NN=16560
19113 -CALL NN      NN=3875
19116 -JR D         D=3 (19121)
19118 -LD (NN),A   NN=16560
19121 -CALL NN      NN=699
19124 -LD A,L
19125 -INC A
19126 -JRNZ D       D=249 (19121)

```

Attend l'appui clavier, passe en FAST, CLS et appelle Emission.

«Recommence»
Sélection MODE «SLOW» et lecture clavier : attend appui C pour continuer le programme

CLS et retour au programme principal

Routine modification vitesse décodage

Lit le clavier si pas d'appui, la vitesse précédente et affichée, sinon on décode l'appui

Touche ↑ (augmenter valeur)

Touche ↓ (diminuer valeur)

Sélectionne le décodage sans affichage et appelle le mode FAST

Attend relâchement de l'appui touche avant de continuer décodage

```

19130-LD HL, (NN)      NN=16396
19131-INC HL
19132-LD A, (NN)      NN=16560
19135-ADD A, N        N=155
19137-LD (HL), A
19138-POP DE
19139-RET

```

Pointe fichier d'affichage, met valeur de vitesse dans l'accu et additionne 128 pour l'afficher en vidéo inversée, au début (1ère colonne, 1ère ligne) de l'écran

Routine de dictée aléatoire

```

19180-CALL NN          NN=2602
19192-LD DE, NN        NN=17796
19195-LD HL, NN        NN=17381
19198-INST IY
19199-253/54/33/64
19200-INST IY          Id (16393), 2
19202-253/54/9/2
19206-INST IY          Id a, (16436)
19206-253/126/52
19209-LD BC, NN       NN=0
19210-LD C, A
19213-PUSH HL
19214-INC BC
19215-LD A, (BC)
19216-INST IY
19216-253/134/52     add a, (16436)
19219-INST IY          Id (16436), a
19219-253/119/52
19220-NOP
19223-CP Z             N=128
19225-LRNC D           D=6 (19233)
19227-CP Z             N=64
19229-LRNC D           D=4 (19235)
19231-JR D             D=5 (19238)
19233-SRL A
19235-SRL A
19237-AND A
19238-CP Z             N=19
19240-LRNC D           D=10 (19252)
19242-CP Z             N=14
19244-LRC D           D=4 (19250)
19246-RLC A
19248-JR D             D=2 (19252)
19250-LD A, N          N=0
19252-LD (DE), A
19253-INC DE
19254-ADD A, N         N=74
19256-LD L, A
19257-LD H, N         N=66
19259-LD A, (HL)
19260-POP HL
19261-LD (HL), A
19262-INC HL
19263-LD (HL), N      N=255
19265-PUSH HL
19266-LD A, N         N=15
19268-RST 16
19269-JR D             D=13 (19264)

```

CLS
Pointe la zone message à afficher
Pointe zone message transcodé

16393 est utilisée comme relai

On utilise FRAMES pour générer un nombre aléatoire

Ce nombre est manipulé pour le rendre exploitable

HL pointe à partir de 16970

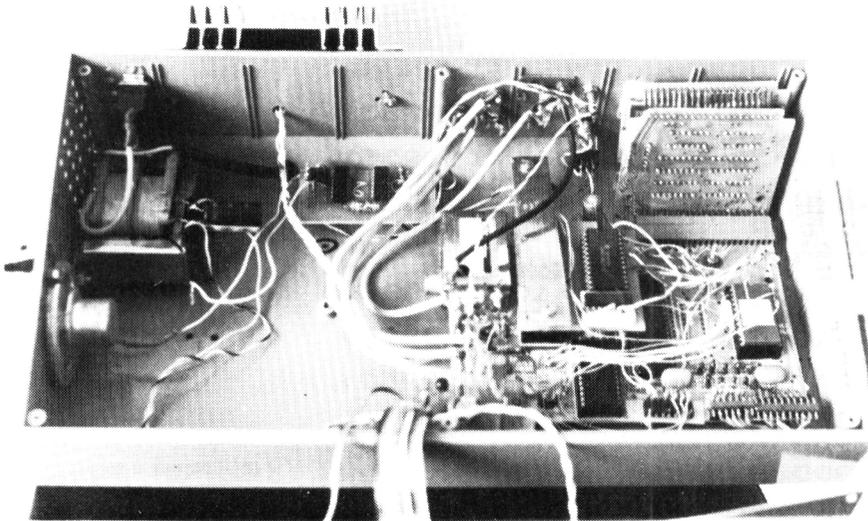
Caractère transcodé vient prendre sa place dans message transcodé

Affiche un ? pour chaque caractère tiré

```

190004 -INST/IY          l d a , (16417)
190004 -253/126/33
190007 -DEC A
190008 -INST/IY          l d (16417) , a
190008 -253/119/33
190091 -LJNZ D           D =177  (19214)
190093 -INST/IY
190093 -253/126/9       l d a , (16393)
190096 -DEC A
190097 -INST/IY          l d (16393) , a
190097 -253/119/9
193000 -LJNZ D           D =163  (19214)
193002 -LD A,N           N =18
193004 -LD (DE),A
193005 -JP NN            NN =16974

```



LE BOITIER ZX MODIFIE, OUVERT.

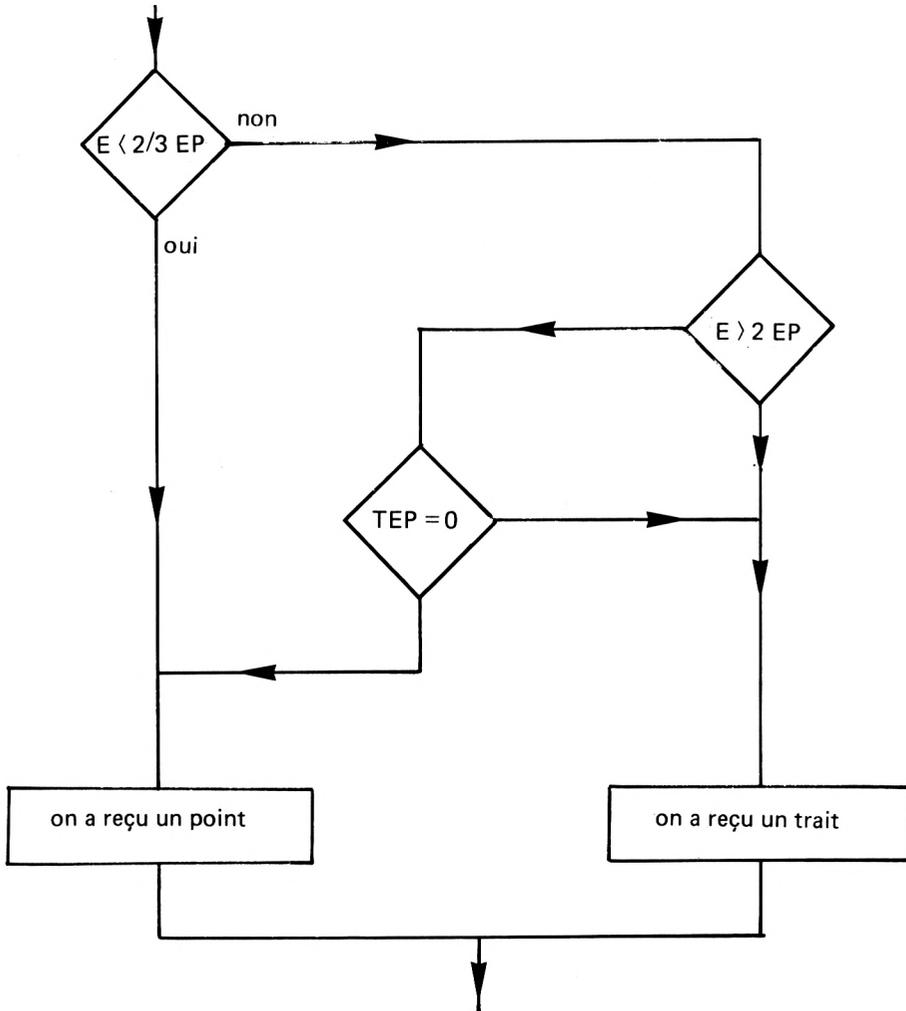
Noter l'alimentation et le HP du bip sonore à gauche, le dissipateur du régulateur à l'arrière.

Principe de reconnaissance TRAIT - POINT

E élément

EP élément précédent

TEP type de l'élément précédent (0 si point, 1 si trait)



Pour un zéro symbolisant un espace, on effectue la comparaison avec la durée du point. Si elle est inférieure à 2 points, on est en présence d'une séparation entre 2 «bits» d'un même élément. Si elle est comprise entre 2 et 4 points, c'est un espace inter-caractères. Supérieure à 4 points, c'est une séparation entre mots.

PROGRAMME QTH AZI

Permet le calcul de la distance entre deux points, connaissant les «QTH-Locators» correspondants. Donne également l'azimut.

Nous rappelons simplement les formules

Distance : (A origine, B arrivée)

$$D = \text{ARCOS} \sin (LA) \sin (LB) + \cos (LA) \cos (LB) \cos (GB - GA) \quad \times 60.$$

Le résultat est en milles nautiques.

Rappelons que le mile nautique (1,852 km) correspond à l'équateur, à un angle d'une minute.

En kilomètres, le facteur 60 deviendra 111.323

L'azimut :

$$AZ = \text{ARCOS} \left[\frac{\sin (LB) - \cos (D/60) \sin (LA)}{\sin (D/60) \cos (LA)} \right]$$

L'azimut est bien sûr en degrés.

Il faut se souvenir que le ZX travaille toujours en mode radians, il faudra donc assurer la transformation (division par PI, multiplication par 180, ou l'inverse).

Les lettres du découpage QTH-Locator correspondent en fait à des secteurs délimités par des méridiens et parallèles. Ainsi, le 0 degré (méridien de Greenwich) passe entre les «carrés» A. et Z. On utilise ces propriétés dans le programme pour effectuer la transformation QTH \Rightarrow position géographique.

On convertira donc les lettres et les chiffres du QTH en valeurs introduites dans une formule les convertissant en latitude et longitude.

Particularités du programme

- Utilisation de la fonction CODE pour déterminer la «valeur» d'une lettre ou d'un chiffre (lignes 70 à 180).
- Utilisation du découpage de chaînes pour effectuer des «décalages» successifs et lire une par une les lettres et chiffres du QTH-Locator (lignes 110, 130, 150, 170).

Voici le mécanisme :

$$\text{Si } Q\$ = BI 23 E$$

ligne 80	A = - 64 + CODE Q\$	donne - 38 + 39 = + 1
ligne 110	Q\$ = Q\$ (2 TO)	donne Q\$ = I 23 E
ligne 120	B = - 38 + CODE Q\$	donne - 38 + 46 = 8
ligne 130	Q\$ = Q\$ (2TO)	donne Q\$ = 23 E
ligne 140	C = - 28 + CODE Q\$	donne - 28 + 30 = - 2
ligne 150	Q\$ = Q\$ (2 TO)	donne Q\$ = 3 E
ligne 160	D = - 28 + CODE Q\$	donne - 28 + 31 = - 3
ligne 170	Q\$ = Q\$ (2 TO)	donne Q\$ = E
ligne 180	E = CODE Q\$	donne 42

Pour retrouver les codes en fonction des chiffres et lettres, reportez-vous à l'annexe A (p. 181) du manuel SINCLAIR. Le mécanisme est simple et les valeurs qui en découlent sont ensuite introduites dans les formules de calcul.

Dernière particularité du programme : les formules sont fractionnées pour rendre plus lisible le listing.

On sauvegarde par GOTO 770 et non SAVE.

```
ENTREZ VOTRE LOCATOR -> BI23E
LES COORDONNEES SONT
-----
LATITUDE :      48.645833
LONGITUDE :      2.5
```

C POUR CONTINUER

```
LOCATOR DU CORRESPONDANT -> ZE19J
LES COORDONNEES SONT
-----
LATITUDE :      44.8125
LONGITUDE :      -0.3
DISTANCE :      477.15337 KMS
-----
AZIMUT :      207.84252 DEG
-----
```

C POUR CONTINUER

Exemple d'exécution du programme QTH AZI

En haut, au lancement du programme, la machine attend l'introduction du QTH-Locator de la station origine. Ici BI 23 E. Elle affiche les coordonnées correspondantes pour information, puis un appui sur une touche démarrera le programme.

En bas figurent les calculs pour le carré ZE 19 J. Noter qu'on aurait pu «tronquer» les décimales ...

```

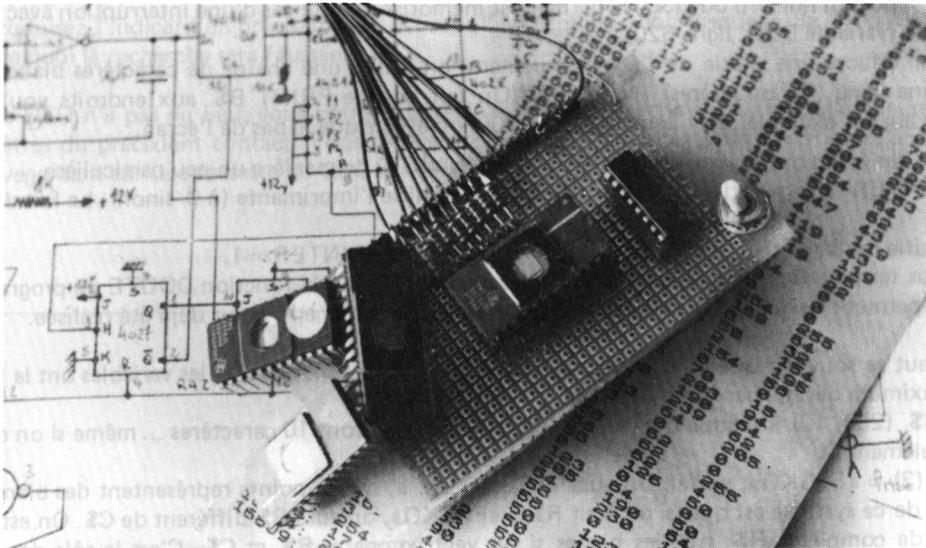
30 REM QTHAZI
40 REM
50 REM F8GK0...1982
50 REM
70 IF CODE Q$ < 57 THEN GOTO 100
80 LET A = -64 + CODE Q$
90 GOTO 110
100 LET A = -38 + CODE Q$
110 LET Q$ = Q$ (2 TO )
120 LET B = -38 + CODE Q$
130 LET Q$ = Q$ (2 TO )
140 LET C = -28 + CODE Q$
150 LET Q$ = Q$ (2 TO )
160 LET D = -28 + CODE Q$
170 LET Q$ = Q$ (2 TO )
180 LET E = CODE Q$
190 IF D < > 0 THEN GOTO 220
200 LET D = 10
210 LET C = C - 1
220 IF E = 38 THEN LET E = 3.1
230 IF E = 39 THEN LET E = 1.1
240 IF E = 40 THEN LET E = 1.3
250 IF E = 41 THEN LET E = 1.5
260 IF E = 42 THEN LET E = 3.5
270 IF E = 43 THEN LET E = 5.5
280 IF E = 44 THEN LET E = 5.3
290 IF E = 45 THEN LET E = 5.1
300 IF E = 47 THEN LET E = 3.3
310 LET H = INT E
320 LET K = ABS ((INT E) - E) * 10
330 LET GB = (2 * A) + (D / 5) - (H / 30)
340 LET LB = 41 + B - (C / 8) - (K / 48)
350 PRINT
360 PRINT "LES COORDONNEES SONT
370 PRINT "-----
380 PRINT "LATITUDE :", LB
390 PRINT
400 PRINT "LONGITUDE :", GB
410 PRINT
420 RETURN
430 PRINT "ENTREZ VOTRE LOCATOR
> " ;
440 INPUT Q$
450 PRINT Q$
460 GOSUB 70
470 LET LA = LB
480 LET GA = GB
490 PRINT AT 21, 7; "C POUR CONTIN

```

```

500 PAUSE 4E4
510 POKE 16437,255
520 CLS
530 PRINT "LOCATOR DU CORRESPON
DANT-> ";
540 INPUT Q$
550 PRINT Q$
560 GOSUB 70
570 LET DG=GA-GB
580 LET A=SIN (LA/180*PI)
590 LET B=SIN (LB/180*PI)
600 LET C=COS (LA/180*PI)
610 LET D=COS (LB/180*PI)
620 LET E=COS (DG/180*PI)
630 LET DIST=111.323*(ACS ((A*B
+(C*D*E)))/PI*180)
640 PRINT "DISTANCE : ",DIST;" K
M"
650 PRINT "-----"
660 PRINT
670 LET DIST=DIST/1.652
680 LET R=DIST/60
690 LET F=COS (R/180*PI)
700 LET G=SIN (R/180*PI)
710 LET AZIMUT=ACS ((B-F*A)/(G*
C))/PI*180
720 PRINT "AZIMUT : ",
730 IF GA-GB>0 THEN LET AZIMUT=
360-AZIMUT
740 PRINT "AZIMUT;" DEG"
750 PRINT "-----"
760 GOTO 490
770 SAVE "0THAZI"
790 FAST
800 GOTO 430

```



CARTE DE PROGRAMMATION DES EPROM.

Noter le compteur d'adresses, en haut l'EPROM sur son support.

PROGRAMME DE CONTEST

On utilise ici la machine comme aide-mémoire, pour gérer le carnet de trafic lors d'un concours. On se limite, à cause des 16 K de mémoire, à 250 QSO. Avec une extension mémoire plus volumineuse, il serait possible de faire mieux.

On passe en revue dans ce programme pratiquement toutes les fonctions principales du ZX.

Dans son module de calcul (lignes 20 à 430), on reprend la partie «distance» du programme QTH AZI, avec ses fonctions trigonométriques et algébriques.

Comme il n'existe pas de fonction «arrondi» sur le ZX, les lignes 410 et 420 effectuent ce calcul, à partir de la fonction INT pour afficher une distance arrondie, sans décimales.

Nous allons examiner les autres particularités :

La faculté de sauvegarder les données et variables en même temps que le programme, a été utilisée pour pouvoir interrompre le contest en cours, sans perdre d'informations. On utilise le SAVE en ligne de programme (1350 à 1450), ce qui provoque l'auto-lancement du programme en fin d'exécution du SAVE. Cela évite le risque de faire RUN, qui détruit tous les fichiers. Ceci est d'ailleurs très souvent utilisé par les auteurs....

Ligne 1410, la programmation d'une PAUSE «infinie» (4E4 = 40000) n'est là que pour offrir la possibilité de démarrer en appuyant sur une touche du clavier. On a préféré cela à la fonction INKEY\$ en pensant aux utilisateurs du ZX 80 ...

- Pour les réponses par OUI ou par NON, on offre la possibilité d'écrire O ou OUI, N ou NON, en effectuant le test seulement sur la première lettre (ligne 510 : CODE R\$).
- On a créé 2 fichiers séparés pour les indicatifs C\$ et les données D\$. Ils sont associés, pour chaque QSO par le numéro du QSO (lignes 570 et 580).
- A propos du numéro du QSO, le dernier est mémorisé dans le cas d'une interruption avec sauvegarde (variable DER, ligne 620).
- Pour effacer une partie de l'écran seulement, on utilise une chaîne de caractères blancs égale à une ligne (32 caractères) (ligne 590). On fait ensuite PRINT B\$ aux endroits voulus sur l'écran. Par exemple : aux lignes 1310 à 1330, on efface que le bas de l'écran.
- Une des fonctions logiques du ZX est utilisée ligne 810 de manière un peu particulière.
PRINTER est une variable positionnée à 1 si on utilise l'imprimante (à 0 sinon). Le fait d'écrire
IF PRINTER THEN ...
signifie implicitement d'exécuter l'action si la variable PRINTER = 1.
- Nous terminerons avec un examen un peu plus détaillé de la fonction DOUTE du programme, qui permet de signaler à un opérateur, s'il le demande, qu'une liaison a déjà été réalisée.

Il faut se souvenir que dans le tableau de chaînes de variables, toutes les variables ont la longueur maximum définie par le tableau.

DIM C\$ (250, 10) implique que toutes les chaînes C\$ auront 10 caractères ... même si on en utilise seulement 5.

Si C\$ (3) = «F6GKQ», en fait on aura «F6GKQ », où les points représentent des blancs. Le défaut de ce système est que, si on écrit R\$ = «F6GKQ», on aura R\$ différent de C\$. On est donc obligé de compléter R\$ par des blancs si on veut comparer R\$ et C\$. C'est le rôle des lignes 1060 à 1090.

L'examen du programme est terminé. Aux lignes 320 et 330 il faut introduire, en degrés décimaux, les coordonnées de votre station. Le programme QTH AZI peut vous les fournir, sinon une autre solution existe à utiliser le programme CONTEST momentanément modifié.

- mettre une ligne 315 STOP
- écrire en mode commande, c'est-à-dire sans numéro de ligne :
 - votre QTH LOCATOR (puis new-line)
 - GOTO 30 (new-line)
 - PRINT LB, GB (new-line)
- vous reportez vos coordonnées aux lignes 320 et 330 et vous ôtez la ligne 315.

Voyons le mode d'emploi du programme :

Quand la machine affiche INDIC (L LISTE D DOUTE S STOP), vous avez plusieurs possibilités :

- vous répondez par un indicatif (9 caractères maximum). La machine affiche alors QTR QRA MON SON. Vous lui donnez dans l'ordre, séparées par un blanc les 4 informations :
 - HEURE (4 chiffres, exemple 0723)
 - QRA (exemple : ZE 19 J)
 - Groupe reçu (exemple : 53021)
 - Groupe passé (exemple : 52012)

puis vous envoyez sur NEW-LINE. Le ZX effectue les calculs distance et points, procède à l'affichage et renvoie la question : INDIC (LISTE D DOUTE S STOP).

- Vous répondez L.

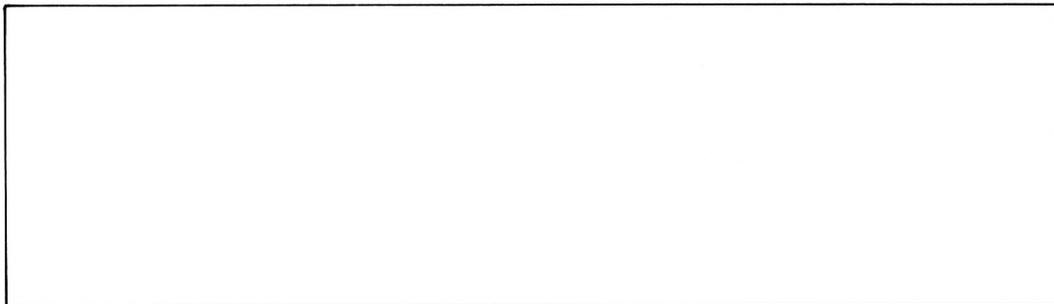
Vous pouvez alors lister les contacts entre 2 numéros. Lorsque l'écran est plein (compteur d'erreur 5), Faire CONT et NEW-LINE.

- Vous répondez D.

Vous avez un doute sur un indicatif et pensez avoir déjà réalisé le QSO. Le ZX écrit : INDICATIF EN DOUTE ?

Introduisez l'indicatif en précisant bien (si vous l'avez toujours fait jusque là) le /P ou /35, etc... faute de quoi la recherche sera faussée.

Si le QSO n'a pas eu lieu, vous pourrez le réaliser, sinon la machine vous rappelle l'heure et les paramètres du précédent contact. Vous répondez par S. Vous pouvez alors interrompre le contest en sauvegardant sur une cassette les QSO réalisés. Le ZX vous indique la procédure.



```

1 REM CONTEST2
2 REM
3 REM F6GK0...1982
4 REM
5 REM PARALLELELOGRAM
600 LET Q#=D#(I) (6 TO 10)
300 IF CODE Q#<57 THEN GOTO 60
400 LET R=-64+CODE Q#
500 GOTO 70
600 LET A=-30+CODE Q#
700 LET Q#=Q#(0 TO )
800 LET B=-30+CODE Q#
900 LET Q#=Q#(0 TO )
1000 LET C=-20+CODE Q#
1100 LET Q#=Q#(0 TO )
1200 LET D=-20+CODE Q#
1300 LET Q#=Q#(0 TO )
1400 LET E=CODE Q#
1500 IF D=0 THEN GOTO 170
1600 GOTO 190
1700 LET D=10
1800 LET C=C-1
1900 IF E=30 THEN LET E=3.1
2000 IF E=30 THEN LET E=1.1
2100 IF E=40 THEN LET E=1.0
2200 IF E=41 THEN LET E=1.5
2300 IF E=42 THEN LET E=0.5
2400 IF E=43 THEN LET E=5.5
2500 IF E=44 THEN LET E=5.0
2600 IF E=45 THEN LET E=5.1
2700 IF E=47 THEN LET E=0.0
2800 LET H=INT E
2900 LET K=ABS ((INT E)-E)*10
3000 LET GB=(2*A)+(D/5)-(H/30)
3100 LET LB=41+B-(C/3)-(K/40)
3200 LET LA=40.645833
3300 LET GD=2.5
3400 LET DG=GD-GB
3500 LET A=5IN (LA/180*PI)
3600 LET B=5IN (LB/180*PI)
3700 LET C=5OS (LA/180*PI)
3800 LET D=5OS (LB/180*PI)
3900 LET E=5OS (DG/180*PI)
4000 LET DIST=111.323*(ACS ((A*B
)+(C*D*E)))/PI*180
410 IF DIST-INT DIST<=.5 THEN L
ET DIST=INT DIST
420 IF DIST-INT DIST>.5 THEN LE
T DIST=INT DIST+1
430 RETURN
440 REM ARTIE CONTEST
450 LET PRINTER=0
460 PRINT "CONTEST EN COURS ?"
470 INPUT A$
480 CLS
490 PRINT "UTILISEREZ-VOUS L""I
MPRIMANTE ?"
500 INPUT R$

```

```

510 IF CODE R$=52 THEN LET PRIN
TER=1
520 CLS
530 IF CODE A$=52 THEN GOTO 610
540 LET CUM=0
550 LET MOY=0
560 LET DER=1
570 DIM C$(250,10)
580 DIM D$(250,22)
590 LET B$=""

600 LET M$="QTR :GRA :MON :SO
4 "
610 PRINT "LE PROCHAIN QSO AURA
LE NO ";DER
620 FOR I=DER TO 150
630 GOSUB 1310
640 PRINT AT 20,0;"INDIC. (L LI
5TE D DOUTE S STOP)"
650 INPUT R$
660 IF R$="S" THEN GOTO 1350
670 IF R$="L" THEN GOTO 870
680 IF R$<>"D" THEN GOTO 710
690 GOSUB 1020
700 IF DOU=1 THEN GOTO 630
710 GOSUB 1310
720 LET C$(I)=R$
730 PRINT AT 20,1;M$
740 PRINT AT 15,0;C$(I);
750 INPUT D$(I)
760 PRINT D$(I)
770 GOSUB 10
780 LET CUM=CUM+DIST
790 LET MOY=CUM/I
800 PRINT "ORB:";TAB 5;DIST;TAB
11;"CUMUL:";CUM;TAB 24;"MOY:";I
4NT MOY
810 IF PRINTER THEN LPRINT C$(I
);D$(I)
820 IF PRINTER THEN LPRINT "ORB
";TAB 5;DIST;TAB 11;"CUMUL:";CU
4;TAB 24;"MOY:";INT MOY
830 GOSUB 1310
840 SCROLL
850 SCROLL
860 NEXT I
870 REM FUNCTION EDITION
880 CLS
890 PRINT (I-1);" QSO REALISES"
900 PRINT
910 PRINT "EDITION A PARTIR DU
40 ?"
920 INPUT N1
930 CLS
940 PRINT "JUSQU""A QUEL NO ?"
950 INPUT N2
960 CLS
970 FOR N=N1 TO N2
980 PRINT C$(N);D$(N)
990 NEXT N
1000 LET DER=I
1010 GOTO 630

```

```

1020 REM EN CAS DE DOUTE
1030 GOSUB 1310
1040 PRINT AT 17,0;"INDICATIF EN
1050 INPUT R$
1060 LET L$=""
1070 LET L=LEN R$
1080 LET U#=L$(L+1 TO 10)
1090 LET R#=R#+U$
1100 LET C$(I)=R$
1110 FOR K=1 TO I-1
1120 IF C$(I)<>C$(K) THEN GOTO 1
1130 GOSUB 1310
1140 PRINT AT 18,0;C$(I);"
1150 PRINT AT 19,0;C$(K);D$(K)
1160 PRINT "APPUI SUR NL POUR CO
1170 INPUT R$
1180 GOSUB 1310
1190 LET DOU=1
1200 GOTO 1300
1210 NEXT K
1220 GOSUB 1310
1230 PRINT AT 18,0;C$(I);"
1240 PRINT "VOULEZ-VOUS CONTACTE
1250 INPUT R$
1260 IF CODE R#=52 THEN GOTO 129
1270 LET DOU=1
1280 GOTO 1300
1290 LET DOU=0
1300 RETURN
1310 FOR M=17 TO 20
1320 PRINT AT M,0;B$
1330 NEXT M
1340 RETURN
1350 REM
1360 CLS
1370 PRINT "SAUVEGARDE DES Q50 E
1380 PRINT "
1390 PRINT
1400 PRINT "POSITIONNER K7 ET PR
1410 PAUSE 4E4
1420 POKE 16437,255
1430 SAVE "CONTEST2"
1440 PRINT AT 15,10;"A BIENTOT..
1450 STOP
1460 SAVE "CONTEST2"
1470 FAST
1480 GOTO 440

```

PROGRAMME DE CALCULS

Nous vous présentons ici le programme «CALCULS 2» car il est conçu de manière «modulaire», ce qui permet d'y ajouter des fonctions. Il suffit pour cela de modifier le «menu» (lignes 2000 à 2140) pour y inclure des options supplémentaires telles que par exemple, les calculs de filtres, le gain d'un ampli, les longueurs d'éléments d'une antenne, etc...

Ces options renvoient par un GOTO sur les petits modules des programmes correspondants. Il suffit donc de créer, autour de la formule considérée, le module de calcul.

Nous ne détaillerons donc pas ce programme puisqu'il ne s'agit en fait que de l'application de formules mathématiques. Notons simplement l'attente de l'appui sur une touche, ligne 2100, qui utilise la fonction INKEY\$ en rebouclant sur le même numéro de ligne tant qu'il n'y a pas eu d'appui.

Ligne 2250, remarquer l'astuce utilisée pour saisir le dernier caractère du nombre introduit (dont on ignore la longueur), qui est la lettre indiquant la base.

En fait (LEN A\$ TO), revient à prendre le dernier caractère de la chaîne A\$.

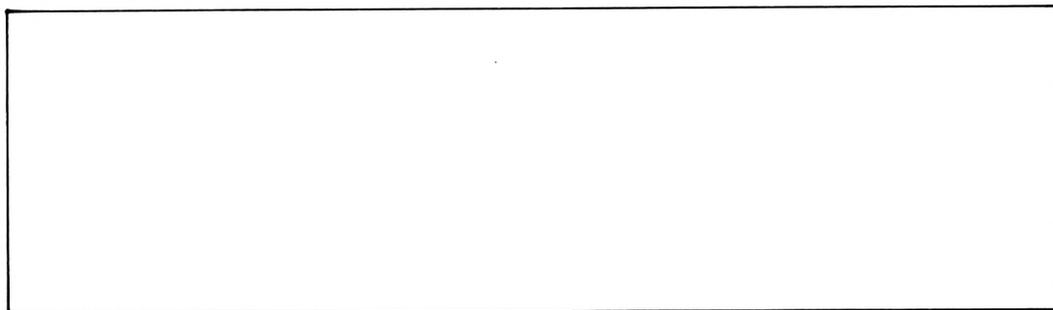
Lignes 2660 à 2720, il y a une routine de détection d'erreurs traitant les caractères du nombre entré et vérifiant s'ils sont compatibles avec la base indiquée. Le fait d'écrire 1A52D provoque «erreur sur le 2ième chiffre» (il ne peut pas y avoir de lettre dans un nombre décimal...).

Noter aussi l'emploi de N * N plus rapide pour les petites puissances que l'utilisation d'un exposant.

On pourrait, bien sûr, exécuter tout ce programme en mode FAST (ajouter une ligne FAST entre 4000 et 4010), mais il faut penser à modifier la saisie de l'option au menu, car la fonction INKEY\$ = " ", rebouclée sur elle-même, ne permet pas l'affichage. Ecrire à la place une ligne :

2100 PAUSE 4E4

suivie d'une ligne 2105 POKE 16437, 255 si vous avez un ZX 80 modifié 81 avec la ROM 8 K ou si vous possédez l'un des tous premiers ZX 81.



PROGRAMME DE CALCULS FIEZH/F6GK0

OPTIONS

- CHANGEMENTS DE BASES
- CALCULS D'ATTENUATEURS
- CALCULS DE SELFS

CHOISISSEZ VOTRE OPTION.

```
1000 REM CALCULS2...FIEZH/F6GK0
2000 CLS
2010 PRINT "PROGRAMME DE CALCULS
2020 PRINT "EN BINAIRE/F6GK0"
2030 PRINT
2040 PRINT
2050 PRINT TAB 12;"OPTIONS"
2060 PRINT
2070 PRINT " CHANGEMENTS DE BA
2080 PRINT "SE"
2090 PRINT " CALCULS D'ATTENU
2100 PRINT "ATEURS"
2110 PRINT " CALCULS DE SELFS"
2120 PRINT AT 20,0;"CHOISISSEZ
2130 PRINT "VOTRE OPTION."
2140 IF INKEY#="" THEN GOTO 2100
2150 IF INKEY#="1" THEN GOTO 215
2160 IF INKEY#="2" THEN GOTO 256
2170 IF INKEY#="3" THEN GOTO 336
2180
2190 STOP
2200 CLS
2210 PRINT TAB 10;"CONVERSION"
2220 PRINT TAB 10;"-----"
2230 PRINT
2240 PRINT "EN BINAIRE, OCTAL, DEC
2250 PRINT "IMAL, HEXA."
2260 PRINT AT 19,0;"ENTRER LE NO
2270 PRINT "MBRE SUIVI DE B,O,D,H SUIVANT LE
2280 PRINT "CAS (R POUR MENU)"
2290 INPUT A$
2300 IF A#="R" THEN GOTO 2000
2310 CLS
```

```

000040 PRINT A#
000050 LET T#=R#((LEN R#) TO )
000060 IF T#="B" OR T#="O" OR T#="
000070 OR T#="H" THEN GOTO 2290
000080 CLS
000090 GOTO 2000
000100 IF T#="O" THEN GOTO 2330
000110 IF T#="O" THEN GOTO 2370
000120 IF T#="B" THEN GOTO 2400
000130 IF T#="H" THEN GOTO 2470
000140 LET U#="9"
000150 GOSUB 2060
000160 LET D=CODE A#(1 TO LEN A#-1)
000170 GOTO 20510
000180 LET U#="7"
000190 GOSUB 2060
000200 LET T1=0
000210 GOSUB 2070
000220 GOTO 20510
000230 LET U#="1"
000240 GOSUB 2060
000250 LET T1=2
000260 GOSUB 2070
000270 GOTO 20510
000280 LET U#="7"
000290 GOSUB 2060
000300 LET T1=15
000310 GOSUB 2070
000320 LET T1=2
000330 GOSUB 2080
000340 LET B#=R#
000350 LET T1=0
000360 GOSUB 2080
000370 LET O#=R#
000380 LET T1=15
000390 GOSUB 2080
000400 LET H#=R#
000410 PRINT
000420 PRINT "BINAIRE.....":B#
000430 PRINT "OCTAL.....":O#
000440 PRINT "DECIMAL.....":D#
000450 PRINT "HEXADECIMALE.....":H#
000460 GOTO 2200
000470 LET T=0
000480 FOR I=1 TO (LEN A#)-1
000490 LET B#=A#(I)
000500 IF B#<"0" OR B#>U# THEN LET
T=I
000510 NEXT I
000520 IF T<>0 THEN PRINT "ERREUR
SUR LE ";T;"IEME CHIFFRE"
000530 RETURN
000540 LET D=0
000550 FOR I=1 TO LEN A#-1
000560 LET B#=A#(I TO (I+1))
000570 IF B#<="9" THEN GOTO 2790
000580 LET B=CODE B#-28
000590 GOTO 2800
000600 LET B=CODE B#-28
000610 LET D=D*T1+B
000620 NEXT I
000630 RETURN

```

```

0030 LET R1=0
0040 LET R#=""
0050 LET R2=INT (R1/T1)
0060 LET R3=R1-T1*R2
0070 IF R3<=9 THEN GOTO 2910
0080 LET R3=R3+20
0090 LET U#=CHR# R3
0100 GOTO 2920
0110 LET U#=STR# R3
0120 LET R#=U#+R#
0130 IF R3<=9 THEN RETURN
0140 LET R1=R2
0150 GOTO 2850
0160 REM PIPOLES ATTENUATEURS
0170 CLS
0180 PRINT AT 0,4;"***** ATTENUA
TEURS *****"
0190 PRINT
0200 PRINT "CIRCUIT EN PI (P) OU
EN TE (T) ?"
0210 INPUT R#
0220 PRINT
0230 PRINT "ATTENUATION DESIREE
:"
0240 INPUT A
0250 PRINT A
0260 LET N=EXP ((A*2.302585)/20)
0270 PRINT
0280 PRINT "IMPEDANCE CARACTERIS
TIQUE ?"
0290 INPUT ZC
0300 PRINT ZC
0310 PRINT
0320 PRINT "POUR ATTENUATION DE
A:" DB SOUS ";ZC;" OHMS,CIRC
UIT EN ";
0330 IF R#="T" THEN GOTO 3230
0340 PRINT "PI IL FAUT:"
0350 PRINT
0360 LET Z1=ZC*((N*N)-1)/(2*N)
0370 LET Z2=ZC*((N+1)/(N-1))
0380 LET Z1=INT (Z1*10)/10
0390 LET Z2=INT (Z2*10)/10
0400 PRINT "2 RESISTANCES DE ";Z
1:" OHMS"
0410 PRINT "1 RESISTANCE DE ";Z
2:" OHMS"
0420 GOTO 3310
0430 PRINT "TE IL FAUT : "
0440 PRINT
0450 LET Z3=ZC*((N-1)/(N+1))
0460 LET Z4=ZC*((2*N)/((N*N)-1))
0470 LET Z3=INT (Z3*10)/10
0480 LET Z4=INT (Z4*10)/10
0490 PRINT "2 RESISTANCES DE ";Z
3:" OHMS"
0500 PRINT "1 RESISTANCE DE ";Z
4:" OHMS"
0510 PRINT
0520 PRINT AT 19,0;"AUTRES VALEU
RS A CALCULER (O/N)?"
0530 INPUT R#
0540 IF R#="N" THEN GOTO 2000
0550 GOTO 2970

```

```

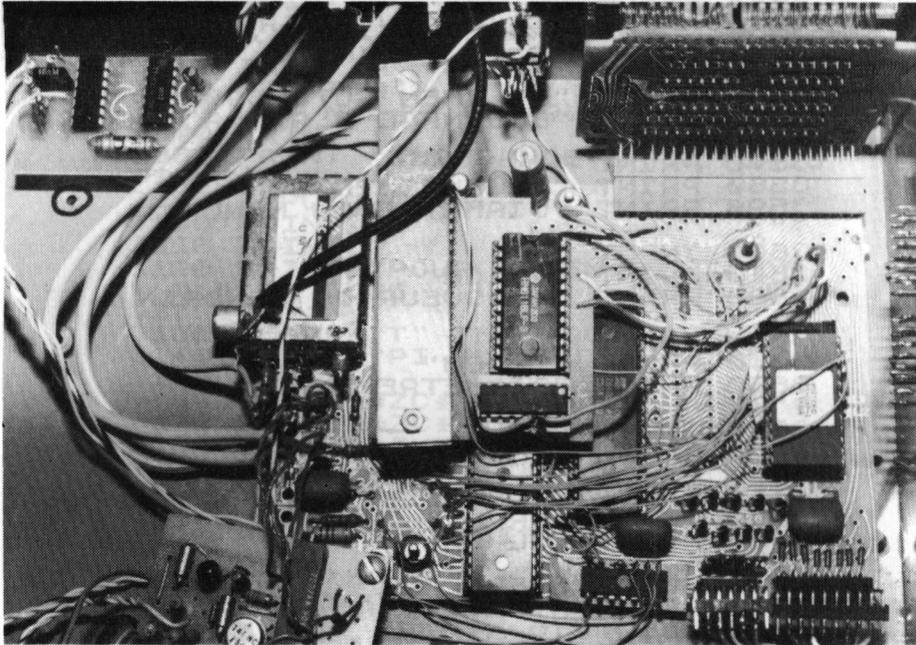
3360 REM CALCULS SELF
3370 CLS
3380 PRINT "**** SELF D""UN CIRC
UIT LC ****"
3390 PRINT
3400 PRINT AT 10,0;"VALEUR DE LA
CAPACITE ?(FARAD)"
3410 INPUT C
3420 PRINT AT 10,0;"FREQUENCE D"
"ACCORD ?(HERTZ)"
3430 INPUT F
3440 LET L=1/(4*PI*PI*F*F*C)
3450 CLS
3460 PRINT "LA VALEUR DE LA SELF
EST ."
3470 PRINT "=====
=====
3480 PRINT
3490 PRINT L;" HENRY"
3500 PRINT
3510 PRINT "=====
=====
3520 PRINT
3530 PRINT "REALISATION: ","-A RI
R(A) " "-DANS UN POT (P)"
3540 INPUT A$
3550 IF A$="P" THEN GOTO 3900
3560 PRINT "BOBINAGE A AIR A 1 C
OUCHE"
3570 PRINT
3580 PRINT "VALEURS EN CM"
3590 PRINT "DIAMETRE DU MANDRIN
?":
3600 INPUT D
3610 PRINT D;" CM"
3620 PRINT "LONGUEUR DU MANDRIN
?":
3630 INPUT I
3640 PRINT I;" CM"
3650 PRINT "DIAMETRE DU FIL
?":
3660 INPUT E
3670 PRINT E;" CM"
3680 LET N=((L*1E5)*((3*D)+(9*I)
+(10*E)))/(0.08*(D**2))
3690 LET N=SQR N
3700 PRINT "IL FAUT ":"N;" SPIRE
S"
3710 GOSUB 3760
3720 PRINT AT 20,0;"AUTRE SELF A
CALCULER ?(O/N)"
3730 INPUT R$
3740 IF R$<>"O" THEN GOTO 2000
3750 GOTO 3360
3760 SCROLL
3770 SCROLL
3780 SCROLL
3790 SCROLL
3800 SCROLL
3810 PRINT "L (HENRY) = ";L
3820 SCROLL
3830 PRINT "C (FARAD) = ";C
3840 SCROLL
3850 PRINT "F (HERTZ) = ";F

```

```

00000 SCROLL
00005 SCROLL
00010 SCROLL
00015 SCROLL
00020 RETURN
00025 PRINT "COEFF. DU POT EN NH/
00030 HRE @ ?";
00035 INPUT A
00040 PRINT A
00045 LET N=INT SQR (L/(A*1E-9))
00050 PRINT "IL FAUT :";N;" SPIRE
00055 GOSUB 3010
00060 GOTO 3700
00070 REM
40000 SAVE "CALCULS2"
40010 GOTO 2000

```



LA CARTE ZX ABONDAMMENT MODIFIEE.

NOTER:

- en haut à gauche, le bip sonore.
- en haut à droite, le 16 K hors de son boîtier.
- à gauche de la 16 K, le poussoir RESET.
- le dissipateur sur le circuit spécialisé.
- la RAM 1 K, 4118 connectée à gauche de la ROM.
- en bas à gauche, l'indicateur de niveau de lecture.
- à côté du connecteur clavier, le 74 LS 32 pattes en l'air.
- la RAM graphique à côté de l'U.L.A.

PROGRAMME "MIRE"

Ce programme permet d'afficher un dessin, très schématisé, ou un indicatif sur l'écran pour l'envoyer sur un émetteur de télévision (si vous avez câblé une prise de sortie VIDEO sur le ZX).

Sa petite particularité est l'affichage de l'heure, mais il faut impérativement être en mode FAST pour que le fonctionnement soit correct. Ce programme est donc exécutable sur ZX 80 8 K ROM.

Le dessin est mémorisé avant la première sauvegarde du programme sur K7, dans les variables M\$ et D\$. Il ne faut donc plus utiliser la commande RUN par la suite.

M\$ contient l'indicatif et D\$ le département. Il est évident que, pour un dessin, on peut supprimer une des deux variables. Avant de sauvegarder pour la première fois le programme, par GOTO 900, il faudra avoir initialisé ces 2 variables.

Pour ce faire, vous taperez FAST, puis RUN et, à l'apparition du curseur «L», vous entrerez votre dessin avec soin. Quand il sera terminé, pressez NEW-LINE. Une seconde fois, le curseur «L» apparaît, correspondant à D\$. Entrez l'autre partie du dessin et NEW-LINE ou, si seule la première partie suffit, directement NEW-LINE.

L'horloge peut être très précise si vous ajustez la PAUSE unité par unité, car elle peut varier d'un ZX à l'autre, très légèrement.

Les dessins sont sauvegardés puisque le SAVE, en ligne de programme, est suivi d'un GOTO 10 et non d'un RUN. Si vous désirez changer, faites RUN.

GTR: 21.07



73"5 A TOUS...

```

001 REM "MIRE"
002 REM
004 REM F6GKO...1981
005 REM
006 INPUT M$
007 INPUT D$
009 PRINT "HEURE ? ";
010 INPUT H
011 PRINT H;" ";
012 PRINT "MINUTES ? ";
013 INPUT M
014 PRINT M
015 PRINT "AU TOP HORAIRE,NEW L
IN 016
017 INPUT T$
018 CLS
019 PRINT
020 PRINT
021 PRINT
022 PRINT M$
023 PRINT
024 PRINT
025 PRINT D$
026 PRINT
027 PRINT
028 PRINT TAB 10;"70""5 A TOUS,
029
030 PRINT AT 0,21;"OTR:";
031 PRINT AT 0,27;
032 IF H=24 THEN LET H=0
033 IF H<10 THEN GOTO 65
034 PRINT H;".";
035 GOTO 70
036 PRINT "0";H;".";
037 IF M<10 THEN GOTO 85
038 PRINT M;
039 GOTO 100
040 PRINT "0";M
041 PAUSE 3005
042 POKE 16437,255
043 LET M=M+1
044 IF M<>60 THEN GOTO 130
045 LET M=0
046 LET H=H+1
047 GOTO 40
048 SAVE "MIRE
049 FAST
050 GOTO 10

```

PROGRAMME QSO TV

Utilise le générateur de caractères, implanté en ROM, pour reproduire à l'écran, en grand format, les lettres et chiffres d'un indicatif.

Ce programme est très lent (en langage machine, son exécution serait beaucoup plus rapide), mais nous le présentons en BASIC car il permet de comprendre simplement comment s'organise le générateur de caractères de la ROM. C'est un peu un développement du programme de démonstration qui a été présenté dans le début de la seconde partie de cet ouvrage.

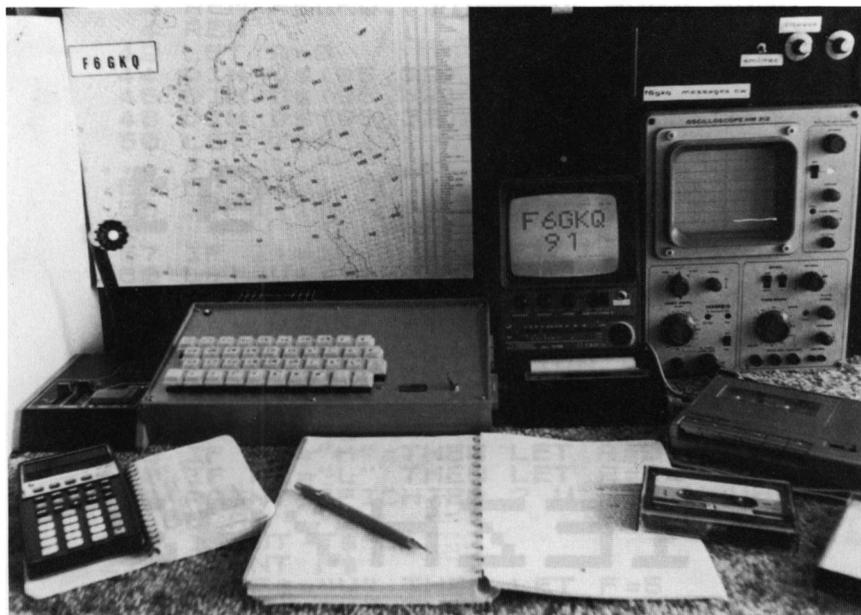
On utilise la fonction PLOT pour allumer les «points» correspondant à un 1 dans le motif binaire du caractère.

Rappelons (ligne 40) que 7680 est l'adresse de la base du générateur de caractères. Chaque caractère étant composé d'une matrice de points sur 8 lignes, pour retrouver l'adresse d'un caractère dans le générateur, on multiplie par 8 le code du caractère et on ajoute au résultat 7680.

La ligne 60 est la ligne clé qui permet de rétablir le profil binaire du caractère.

Le reste du programme appelle peu de commentaires. La variable Z détermine la ligne de l'affichage.

Le programme est en mode FAST (il est bien assez long ainsi), ce qui le rend accessible aux possesseurs de ZX 80 8 K ROM.



L'ENSEMBLE DE TRAVAIL AU COMPLET

```

1 REM "Q50TV"
2 REM
3 REM F6GKQ...1981
4 REM
5 PRINT "INDICATIF DU CORRESP
NDANT ?"
10 INPUT C#
15 CLS
20 GOTO 105
35 FOR M=1 TO LEN J#
40 LET B=7680+CODE J#(M)*8
45 FOR V=B TO B+7
50 LET L=PEEK V
55 FOR F=0 TO 7
60 LET T=L-2**(7-F)
65 IF T<0 THEN GOTO 75
70 LET L=T
71 PLOT X+F,Z-U+B
75 NEXT F
80 NEXT V
85 LET X=X+8
90 NEXT M
95 RETURN
105 LET X=0
110 LET J#="F6GKQ 91"
120 LET Z=40
130 GOSUB 35
140 LET X=0
150 LET J#=C#
160 LET Z=10
170 PRINT AT 11,10;"ESSAI TU AV
C : "
180 GOSUB 35
190 STOP
200000 SAVE "Q50TV"
200001 FAST
200002 RUN

```

F6GKQ 91

ESSAI TU AVEC :

F1EZH/92

PROGRAMME "FIDEP"

Ce programme est un simple fichier de rangement par départements, prévu pour un classement sur 3 bandes des stations contactées, par exemple, dans le but du DDFM.

Ce programme crée 3 fichiers différents qu'il est possible de modifier à sa guise. Après chaque modification il faut, bien sûr, effectuer une nouvelle sauvegarde.

Le programme n'appelle que peu de commentaires. Il faut le lancer par RUN la première fois pour initialiser les fichiers puis le sauvegarder par GOTO 9990.

Ligne 56, vous noterez le traitement de numéro de ligne dans le fichier qui permet d'ajouter 0 devant les chiffres inférieurs à 10, pour une présentation plus homogène. Ensuite, le numéro est concaténé avec la chaîne de caractères représentative du QSO (lignes 60 à 70).

Ligne 280, vous noterez l'utilisation du GOTO calculé, autorisé par le BASIC du ZX.

Pour obtenir une nouvelle page lors du listage, faire CONT, puis NEW-LINE à l'apparition du compte-rendu 5/...

Respectez, lors de l'écriture des QSO, le modèle proposé, pour une présentation agréable à l'écran.

```
1 REM "FIDEP"
2 REM
3 REM F6GKQ....1981
4 REM
42 LET A=0
44 DIM U$(95,32)
46 DIM U$(95,32)
48 DIM D$(95,32)
50 LET Z$="*.....*.....*"
*.....*.....*
55 FOR I=1 TO 95
56 IF I<10 THEN LET I$="0"+STR
# I
57 IF I>=10 THEN LET I$=STR$ I
60 LET U$(I)=I$+Z$
65 LET U$(I)=I$+Z$
70 LET D$(I)=I$+Z$
75 NEXT I
150 PRINT "MODIFICATION DU LIST
AGE ? (M/L)"
155 INPUT A$
160 PRINT A$
165 IF A$="M" THEN LET A=0
170 IF A$="L" THEN LET A=1
200 PRINT "FICHER ? U VHF,U UH
F O DECA"
205 INPUT F$
210 PRINT F$
215 IF F$="U" THEN LET F=5
220 IF F$="U" THEN LET F=6
225 IF F$="D" THEN LET F=7
```

```

230 IF A=1 THEN GOTO 850
250 PRINT "NO DU NOUVEAU DEPART
EMENT ?"
255 INPUT N
260 PRINT N
270 PRINT AT 18,0;"FORMAT D""EC
RITURE A RESPECTER"
275 PRINT AT 19,0;"02.FIBBD....
05/02/80.MQSLB.SQSLD"
280 GOTO F*100
300 REM FICHER UHF
310 PRINT U$(N)
320 INPUT U$(N)
330 PRINT U$(N)
340 GOTO 800
360 REM FICHER UHF
310 PRINT U$(N)
320 INPUT U$(N)
330 PRINT U$(N)
340 GOTO 800
700 REM FICHER DECA
710 PRINT D$(N)
720 INPUT D$(N)
730 PRINT D$(N)
800 SCROLL
801 SCROLL
802 SCROLL
803 PRINT AT 19,0;"VOULEZ-VOUS
LISTER LE FICHER ?"
805 PRINT "REPOSE O/N"
810 INPUT R$
815 PRINT R$
816 CLS
820 IF R$="O" THEN GOTO 850
830 PRINT "AUTRE MODIFICATION ?
(O/N) "
835 INPUT R$
840 PRINT R$
845 IF R$="O" THEN GOTO 150
847 GOTO 900
850 PRINT "LISTAGE A QUEL NO ?"
855 INPUT P
857 CLS
860 FOR I=P TO 95
862 IF F=5 THEN PRINT U$(I)
864 IF F=6 THEN PRINT U$(I)
866 IF F=7 THEN PRINT D$(I)
870 NEXT I
880 GOTO 910
900 CLS
910 PRINT "BAUVEGARDE DU PROGRE
S SUR F"
915 PRINT "DEMARRER MAGNETO ET
FAIRE CONT "
920 PRINT "GOTO 150 POUR UNE AU
TRE OPTION "
1000 STOP
9990 SAVE "FIDE"
9995 GOTO 150

```

10700 OCTETS

PROGRAMME EMISSION-RECEPTION RTTY =====

L'émission-réception RTTY est une chose très intéressante et il est très aisé d'utiliser un micro-ordinateur.

Dans la définition du programme il faut considérer plusieurs paramètres, tels que :

- la vitesse maximum recherchée,
- les possibilités de mémorisation,
- la simplicité des interfaces.

Nous avons choisi une solution qui est un compromis sur plusieurs paramètres. La vitesse maximum sera de 110 bauds. Le shift est programmable à l'émission. Nous aurons la possibilité d'émettre des messages mémorisés. A l'émission, nous n'aurons pas besoin de circuit d'interface. Eventuellement, il faudra prévoir un préampli (1 transistor) entre la sortie SAVE (MIC) du ZX et l'entrée BF (micro) de l'émetteur.

Par contre, en réception, il faudra prévoir des circuits permettant de coupler le récepteur au ZX. Nous avons choisi les plus simples.

Le programme permet en outre quelques possibilités, telles que :

- effaçage de la partie supérieure de l'écran (message décodé),
- forçage en mode chiffres ou lettres,
- copie de l'écran sur imprimante.

Après l'émission d'un message, le retour en réception est automatique.

Le programme permet d'émettre ou non les avances papier et retour chariot, après la nième colonne.

Résumé des possibilités :

- émission de 45,45 à 110 bauds,
- shift normal ou inverse (170, 425, 850 Hz),
- programmation RC et LF,
- messages mémorisés,
- réception selon interface shift programmable.

Il faut noter que l'écran est partagé à l'image du temps de travail de la machine qui décode un message et lit simultanément le clavier, permettant de composer un texte.

Le passage en émission s'effectue à la fin du texte composé, ce qui demande un petit peu d'habitude pour les correspondants car ils doivent toujours attendre un peu à la fin de la frappe du texte ...

Ce défaut est mineur et il évite d'utiliser un générateur AFSK à l'émission. Evidemment, en utilisant la section EMISSION de l'UART, on pourra modifier le programme en conséquence.

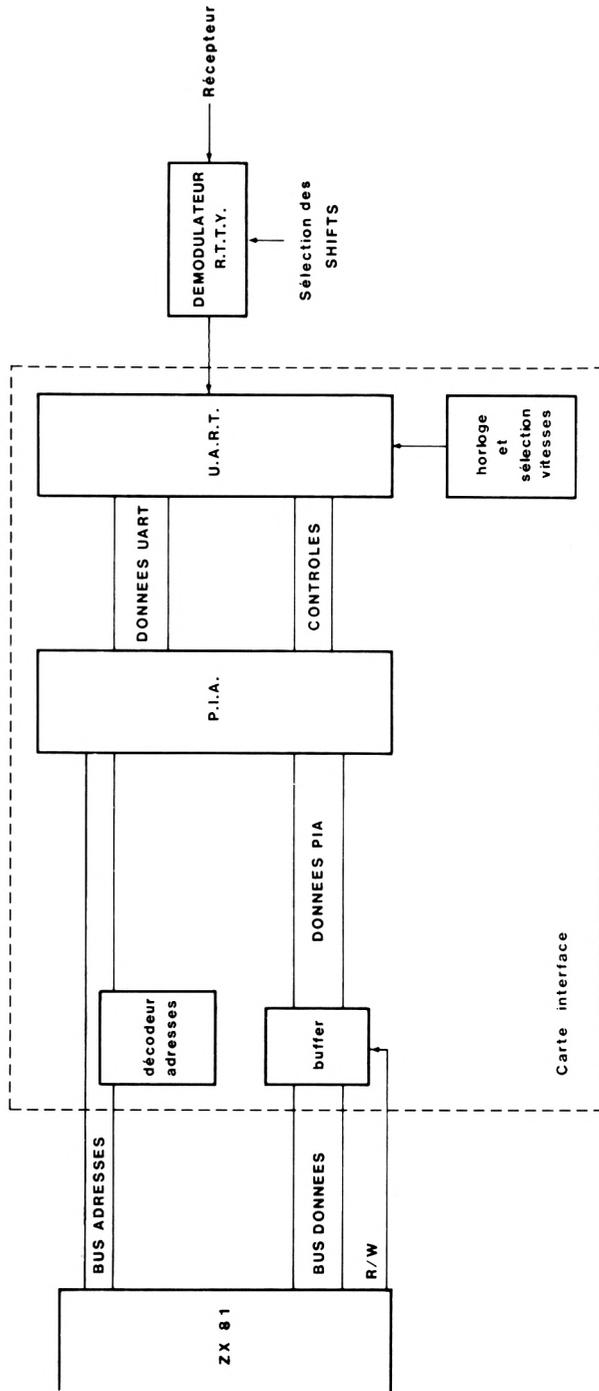


FIGURE 16 : ORGANISATION DU DECODAGE RTTY SUR ZX 81
AVEC AFFICHAGE SIMULTANE.

Les commandes du programme sont :

SHIFT E pour passer en émission,
SHIFT W pour copie sur imprimante,
SHIFT R pour forçage en mode lettres,
SHIFT T pour effacer l'écran,
SHIFT Y pour forçage en mode chiffres.

L'interface à réaliser est la carte PIA décrite plus loin, couplée au circuit UART que nous allons examiner.

Le démodulateur RTTY pourra être quelconque (à filtres ou P.L.L.). Nous vous proposons un montage avec le circuit XR 2211 de EXAR, tiré de la note d'applications.

Un synoptique de l'ensemble est présenté Figure 16.

UTILITE DE L'UART

Il est facile, à l'émission, de générer un message série au format BAUDOT et de produire les tonalités sur la sortie SAVE du ZX en mettant la machine en mode FAST.

Il est possible, à la réception, de décoder un message RTTY en étant également en mode FAST (en mode SLOW, les nombreuses interruptions provoquent une désynchronisation du décodage), mais on y perd en intérêt, puisque l'écran reste éteint pendant le décodage.

Il faut «soulager» le microprocesseur d'une partie du travail. C'est le rôle de l'UART (Universal Asynchronous Receiver Transmitter) qui réalise la transformation du message série (ici en code BAUDOT) en un message parallèle.

Chaque caractère présent sur les sorties «parallèles» de l'UART sera lu par le PIA (peripheral interface adapter) et envoyé vers le micro-ordinateur. Pendant ce temps de décodage et de transformation, le micro-ordinateur peut gérer son écran et lire le clavier.

L'UART est un circuit complexe mais bon marché. Ses possibilités sont multiples et nous ne les utiliserons pas pleinement ici. Les circuits testés sont HD 6402, TR 1863 (mono-tension), AY-5-1013, TMS 6011 (nécessitent — 12 V sur patte 2). En mono-tension il existe aussi le AY-3-1015. Les circuits mono-tension sont plus chers (75,00 F pour AY-3-1015), mais plus pratiques.

L'examen du schéma de l'UART montre sa grande simplicité.

L'horloge est bâtie autour d'un NE555. Les différentes vitesses se font par commutation de résistances. Le réglage fin est obtenu avec des ajustables multi-tours. La fréquence peut être mesurée sur la patte 3 du NE555, au fréquencemètre BF ou à l'oscilloscope (moins précis). Une méthode empirique consiste à enregistrer un message généré par le programme émission, puis à régler l'horloge en le décodant sans erreur.

La sortie Frame Error (FE) de l'UART commande l'allumage d'une diode électroluminescente, avertissant que le format du message reçu est incorrect (codage, mauvais shift, mauvaise vitesse choisie ...).

La réalisation pratique peut s'effectuer sur une plaque pastillée en câblage conventionnel ou en «mini-wrapping», ou sur un circuit imprimé.

Les connexions à réaliser sont :

- l'entrée des signaux venant du démodulateur RTTY par un fil blindé,
- le couplage avec le PIA par de la nappe souple,
- la patte 2 de l'UART sera en l'air ou connectée au — 12 V selon que le composant est mono-tension ou non,
- le + 5 V peut venir du ZX. La masse est connectée à la masse ZX.

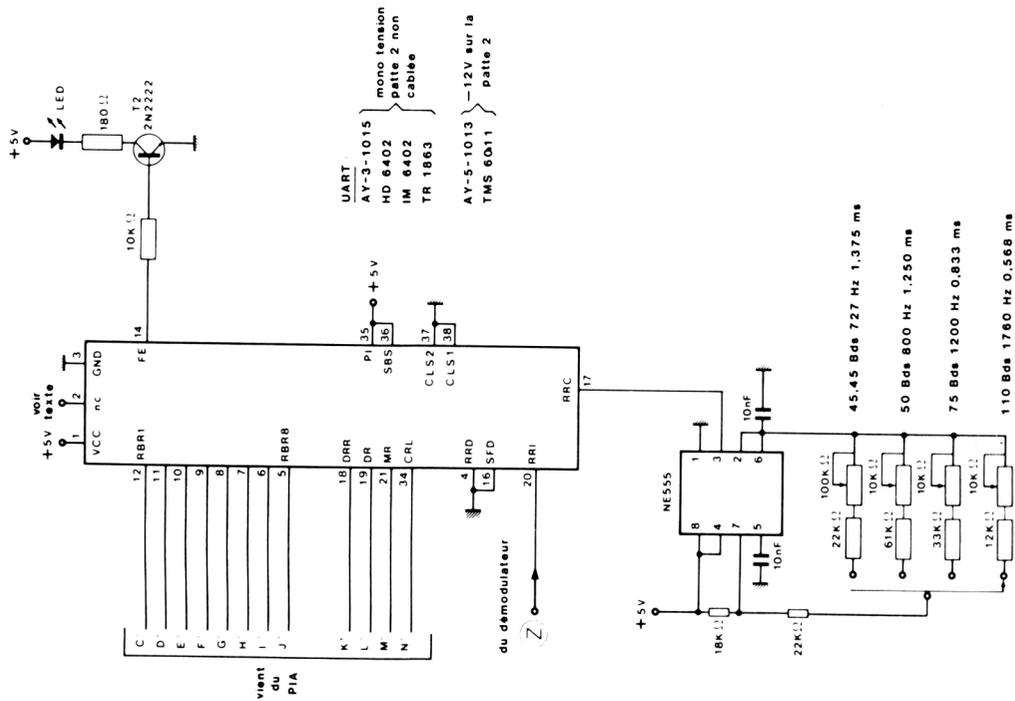


FIGURE 17 : UART ET SON HORLOGE.

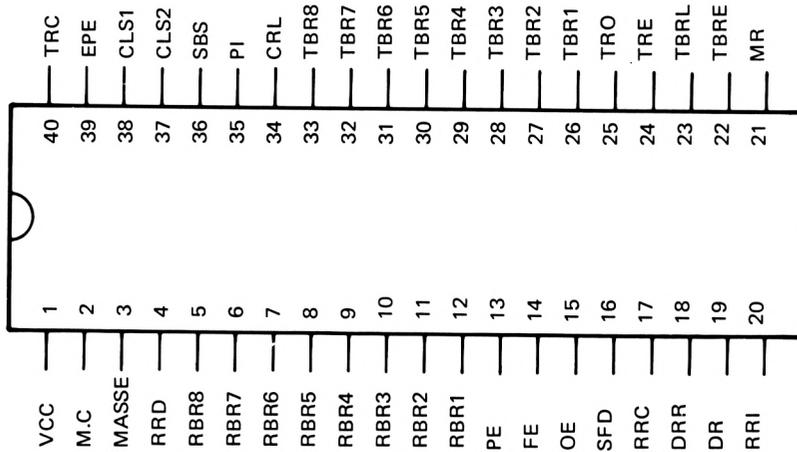


FIGURE 18 : BROCHAGE DE L'UART
HD 6402 (Matra-Harris).

Les brochages sont compatibles
 pour les différents circuits cités.
 Seule la broche 2 est «non con-
 nectée» sur les mono-tensions, au
 - 12 V sinon.

Découpler l'alimentation de la carte par 10 μ F et 10 nF en parallèle.

Le schéma est présenté Figure 17.

Vous trouverez, Figure 18, le brochage complet de l'UART.

Les brochages sont compatibles pour les différents circuits cités. Seule la broche 2 est «non-connectée» sur les mono-tensions, au - 12 V sinon.

Le schéma du démodulateur avec le XR 2211 se suffit à lui-même. Le système de commutation est facultatif (Voir Figure 19).

Procédons à l'examen du programme. Pour mieux le comprendre, reportez-vous à l'organigramme correspondant. En fait, le principe retenu consiste à lire le registre UART le plus fréquemment possible pour être certain de ne jamais manquer un caractère. Tant que le registre UART n'est pas plein, on scrute le clavier pour pouvoir, éventuellement, saisir le texte à émettre. Si aucune touche du clavier n'est sollicitée, on retourne lire l'UART. Dans le cas contraire, on décode la touche et on affiche le caractère correspondant. Si c'est une touche de fonction, on exécute la fonction correspondante et on revient à la lecture de l'UART. Si la touche n'est pas autorisée, on revient également à la lecture UART.

Si la lecture UART indique qu'un caractère est présent, on le décode, on effectue un scrolling de la partie haute de l'écran si nécessaire, et on affiche.

Le listing assembleur est commenté suffisamment.

Ce programme fait appel à des routines originales, telles que le SCROLLING partiel de l'écran ou l'effacement partiel (partie haute de l'écran), mais aussi aux routines déjà écrites dans la ROM, telles que la lecture du clavier.

Une petite astuce a été utilisée pour émettre ou non les retours chariot et sauts de ligne. En effet, selon le choix, on transforme par l'intermédiaire (POKE) du programme BASIC, le programme en langage machine. En 17916 on introduit, soit un DJNZ (16) pour décrémenter le registre B chargé avec le nombre de colonnes, soit un JR (24) si on ne veut pas émettre retours chariot et sauts de ligne.

On voit d'ailleurs que le programme BASIC modifie à plusieurs endroits le langage machine (par exemple pour la vitesse).

Le programme BASIC est très simple. Il permet de définir certains paramètres comme la vitesse, le shift, les messages mémorisés.

Noter le REM en ligne 90 qui contient le texte qui va être imprimé au bas de l'écran, sur les deux lignes «interdites» par un transfert BASIC lignes 7002 à 7009.

Ligne 7250, on transforme par la fonction VAL la chaîne R\$ qui contient le nombre de colonnes pour retour chariot, en valeur numérique avant le POKE.

Ligne 7460, utilisation du GOTO calculé (en fonction de l'option).

Pour les options, on peut, bien sûr, en ajouter en définissant autant de types de messages qu'on le désire ... et que la mémoire le permet.

Il faut leur donner des noms différents. Penser à ajouter au message le curseur qui sert de test de fin de chaîne (cf. par exemple la ligne 8040).

Le programme sera sauvegardé par GOTO 8887.

INTRODUCTION DES LISTES DU LANGAGE MACHINE

Utilisez le programme de création de REM pour créer un espace en première ligne du programme, de 2033 octets. Puis écrire la ligne 90.

PRINT PEEK 18548 doit donner 118.

Par le procédé habituel qui vous est maintenant familier, vous remplirez ce REM avec les octets donnés dans les listes. Soyez vigilants car la moindre erreur peut être fatale. Sauvegardez, après introduction du langage machine, puis écrivez la partie BASIC et sauvegardez-la par GOTO 8887. Dans la définition de vos messages personnels, n'oubliez pas qu'ils ne peuvent pas dépasser 10 lignes de 32 caractères.

UTILISATION DU PROGRAMME

Après chargement l'écran est partagé en deux. Dans la partie supérieure apparaît le message décodé (les interfaces sont, bien sûr, connectées). Dans la partie inférieure, un curseur vous donne la position d'écriture et les options (touches shiftées) vous sont rappelées.

- le message à émettre ne peut dépasser 10 x 32 caractères.
- pour effacer, utiliser la touche shift 5 ; pas de RUBOUT.
- pour émettre, faire shift E. L'écran présente alors l'aspect de la figure 21. Si vous voulez modifier un paramètre, répondez 0. A la fin de la séquence de modifications, la question vous sera posée à nouveau, au cas où vous auriez fait une erreur.

Vous verrez ensuite apparaître les options (Figure 22).

Après avoir fait votre choix, appuyez sur une touche : cela va déclencher l'émission du MARK. Appuyez sur la touche « . » (point), le message est envoyé.

- en fin d'émission le retour en décodage est automatique.
- si, pour une raison ou une autre, vous faites BREAK, il faut relancer par GOTO 7000, non par RUN.

Comme il n'y a pas de générateur d'AFSK, et que votre correspondant peut vous repasser le clavier avant que vous ayez fini de taper votre texte, vous pouvez toujours enclencher votre porteur pour le faire patienter ... s'il n'est pas prévenu (dans le cas de la FM évidemment).

Les figures 20 à 22 vous présentent les différents aspects de l'écran lors du déroulement du programme.

Moyennant quelques modifications du programme lors de l'adressage du PIA, les possesseurs d'une carte de type 8 entrées-sorties pourront l'utiliser.

Notes correspondant à la figure 19.

Commutateur des diverses valeurs de shift :

170 Hz	C1	39nF	R1	150k Ω
425 Hz	C2	33nF	R2	63 k Ω
850 Hz	C3	27nF	R3	36 k Ω

Les inverseurs 1 , 2 , 3 , 4 , 5 , 6 sont ceux d'un CD 4049.

La patte 1 du 4049 au + 5V.

La patte 8 du 4049 à la masse.

- Le commutateur C1 1A / C1 1B peut être omis si on ne veut recevoir que des émissions amateurs au shift 170 Hz (de même que C2, C3, R2, R3).
- La tension admissible à l'entrée va de 2 mV. à 3V. eff. Si vous vous branchez sur une sortie HP, mettez un capa de liaison et des diodes tête-bêche en protection.
- D1 et D3 clignotent au rythme MARK / SPACE.
- D2 doit s'éteindre sur un bon calage.
- Le seul réglage consiste à ajuster la fréquence centrale du VCO.

Configurations de l'écran en RTTY

TRE PARM NAR LA MABS C "EST TOU
T. DE PLUS ON L"ENTEND JURER EN
PHONIE EN SURIMPRESSION QUELLE
EST CETTE NOUVEAUTE DANS LA BID
QUILLE????????? C"ESA PEUT ETR
E POUR CELA QUE NOUS N"ARRIVONS
PAS A DECODER. SI TU MODULE EN
MEME TEMPS. ICI JE N"AI PRATIQUEM
ENT PAS PRIS DE RETOUR RETOUR
CHARIOT NI D"AVANCE PAPIER.TA

VOICI, PRIS SUR LE VIF, UN TEXTE
DECODE TOUT EN COMPOSANT LE
TEXTE DU BAS DE L"ECRAN.....

Fig. 20 : en réception

EMISSIION

PARAMETRES EN MEMOIRE

1- ~~LITESSE~~ 45 BDS
2- ~~SHIFT~~ 170 HZ ~~NOB~~
3- ~~BANS EMISSION DU R.C.~~

--- MODIFICATIONS PARAMETRES LOAN ---

VOICI, CI-DESSUS, LA PRESENTATION
DE L"ECRAN AVANT LE PASSAGE EN
EMISSIION, AVEC LES DIFFERENTES
OPTIONS PROPOSEES. LA PARTIE IN-
FERIEURE AFFICHE QUANT A ELLE LE
TEXTE PREPARE POUR L"EMISSIION...
CE TEXTE EST SUIVI DU CURSEUR
D"ECRITURE... ■

Fig. 21 : en émission
(paramètres)

VOULEZ-VOUS EMETTRE :

- 1-LE MESSAGE AFFICHE 3
- 2-LE MESSAGE MEMORISE 3
- 3-LE MESSAGE DE TEST 3
- 4-LE MESSAGE D'APPEL 3

```
APPUYER SUR UNE TOUCHE
-----
C0 C0 C0 DE F6GK0 F6GK0 F6GK0
C0 C0 C0 DE F6GK0 F6GK0 F6GK0
C0 C0 C0 DE F6GK0 F6GK0 F6GK0
NTH CORBEIL NEAR PARIS (91)
NAME DENIS DENIS DENIS
C0 C0 C0 DE F6GK0 F6GK0 F6GK0
C0 C0 C0 DE F6GK0 F6GK0 F6GK0
C0 C0 C0 DE F6GK0 F6GK0 F6GK0
PSE K K K .....
```

Fig. 22 : en émission
(choix du message)

```
90 REM EMISSION
95 REM FIEZH-F6GK0
99 REM TTY.V01.30.01.83
7000 PRINT AT 11,0; "-----"
7001 PRINT AT 12,0; "=="
7002 LET DF=(PEEK 16396+256*PEEK
16397)+726
7003 FOR I=1 TO 32
7004 POKE (DF+I),PEEK (18554+I)
7005 NEXT I
7006 LET DF=DF+33
7007 FOR I=1 TO 32
7008 POKE (DF+I),PEEK (18586+I)
7009 NEXT I
7010 SLOW
7011 RAND USR 17210
7012 RAND USR 17940
7015 PRINT AT 0,11;"EMISSION"
7020 PRINT
7025 PRINT TAB 5;"PARAMETRES EN
MEDIUM"
7030 PRINT
7035 PRINT "1-VITESSE",U;" ";TA
B 23;"BDS"
7040 PRINT
7045 PRINT "2-SHIFT",ABS B;TAB
23;"HZ";
7047 IF B<0 THEN PRINT TAB 27;"E"
7049 IF B>0 THEN PRINT TAB 27;"E"
7050 PRINT
```

```

7060 IF PEEK 17916=16 THEN PRINT
"0-RECHERCHER APPEL",PEEK 17857;"
";TAB 23;"COLONNES"
7065 IF PEEK 17916=24 THEN PRINT
"0-SAISIR L'EMISSIION DU R.C."
7100 PRINT
7150 PRINT "--INDICATIONS PEEK
16514,0"
7155 IF INKEY$="" THEN GOTO 7155
7160 IF INKEY$="O" THEN GOTO 717
0
7162 IF INKEY$="N" THEN GOTO 738
3
7165 GOTO 7155
7170 REM MODIF DES PARAM
7175 PRINT AT 10,0;"SHIFT: 170,4
25,850(-6I-INVERSE)"
7180 INPUT B
7205 PRINT AT 10,0;"VITESSE:45,5
0,75,110"
7210 INPUT U
7215 LET C=INT ((1/U)*(1275+ABS
B*(B<0)))
7220 POKE 16514,C
7225 LET C=INT ((1/U)*(1275+ABS
B*(B>0)))
7230 POKE 16516,C
7235 PRINT AT 10,0;"@ OU NB. DE
COLONNES POUR R.C."
7240 INPUT R$
7245 IF R$(1)="@" THEN GOTO 7265
7250 POKE 17857,VAL R$
7255 POKE 17916,16
7260 GOTO 7270
7265 POKE 17916,24
7270 IF B<0 THEN GOTO 7350
7275 POKE 16515,91
7280 IF B=170 THEN POKE 16517,80
7285 IF B=425 THEN POKE 16517,67
7290 IF B=850 THEN POKE 16517,52
7295 GOTO 7015
7350 POKE 16517,91
7355 IF B=-170 THEN POKE 16515,8
0
7360 IF B=-425 THEN POKE 16515,6
7
7365 IF B=-850 THEN POKE 16515,5
E
7370 GOTO 7015
7380 RAND USR 17940
7390 PRINT AT 11,0;"-----
-----"
7400 PRINT AT 0,0;"VOULEZ-VOUS E
METTRE:"
7405 PRINT
7410 PRINT "1-LE MESSAGE AFFICHE
?"
7415 PRINT "2-LE MESSAGE MEMORIS
?"
7420 PRINT "3-LE MESSAGE DE TEST
?"
7425 PRINT "4-LE MESSAGE D'"APPE
L
?"

```

```

7435 PRINT
7440 PRINT AT 10,0:"VOTRE CHOIX"
7450 IF INKEY#="" THEN GOTO 7450
7460 GOTO 8000+(VAL INKEY#)*10
8010 GOTO 8400
8020 PRINT AT 12,0;M#;"="
8030 GOTO 8400
8030 PRINT AT 12,0;T#;"="
8035 GOTO 8400
8040 PRINT AT 12,0;A#;"="
8400 REM EMISSION
8410 RAND USR 17841
8505 PRINT AT 10,0:"APPUYEZ SUR"
8510 IF INKEY#="" THEN GOTO 8510
8515 CLS
8520 FAST
8530 RAND USR 16518
8550 GOTO 7000
8807 SAVE "TTY"
8808 REM INITIALISE MESSAGES
8800 LET A#="CQ CQ CQ DE F6GKQ F
F6GKQ F6GKQ CQ CQ CQ DE F6GKQ F
F6GKQ F6GKQ CQ CQ CQ DE F6GKQ F
F6GKQ F6GKQ QTH CORBEIL NEAR PA
RIS (91) NAME DENIS DENIS DE
NIS CQ CQ CQ DE F6GKQ
F6GKQ F6GKQ CQ CQ CQ DE F6GKQ
F6GKQ F6GKQ CQ CQ CQ DE F6GKQ
F6GKQ F6GKQ PSE K K K ....."
8905 LET T#="F6GKQ F6GKQ F6GKQ F
F6GKQ F6GKQ TEST TRANSMISSION
Y COMPUTER ZX 81 SOFTWARE F1EZ
I-F6GKQ
-----QWERTYUIOPASDFGHJKL
ZXCVBNM 1234567890$() -+=:;
?/*<>.,.
-----F6GKQ F6GKQ F6GKQ F
F6GKQ F6GKQ PSE K K K ....."
9010 LET M#="F6GKQ F6GKQ F6GKQ F
F6GKQ F6GKQ NAME DENIS DENIS Q
T NR PARIS C O R B E I L (DPT
91) (BI 20E) COMPUTER ZX 81 SOFT
WARE BY F1EZH-F6GKQ (HARD:U
ART + PIA) UHF:MULTI 750E + 14
EL AT 22 M SHORTWAVES:ATLAS RX
110 DIPOLE F6GKQ DENIS CORBEIL
NEAR PARIS PSE SK ....."
9050 GOTO 7000

```

A cette liste du programme, AJOUTER :

```

8900 LET V = 45.45
8910 LET B = -170

```

165514	:	05	01	00	00	00
165519	:	4	011	00	00	00
165524	:	04	71	00	00	00
165529	:	70	25	00	00	00
165534	:	71	16	00	00	00
165539	:	01	01	00	00	00
165544	:	0	0	00	00	00
165549	:	000	0	00	00	00
165554	:	000	100	00	00	00
165559	:	110	005	7	00	00
165564	:	5	01	00	00	00
165569	:	0010	7	00	00	00
165574	:	0040	107	00	00	00
165579	:	000	100	04	111	01
165584	:	100	111	00	4	011
165589	:	000	00	101	04	71
165594	:	10	004	00	05	15
165599	:	00	101	04	71	16
165604	:	004	45	00	004	100
165609	:	0	04	100	40	000
165614	:	001	7	040	107	00
165619	:	100	04	111	00	4
165624	:	011	005	00	101	04
165629	:	71	16	004	000	70
165634	:	15	00	101	04	71
165639	:	10	004	45	00	004
165644	:	100	041	001	040	1007
165649	:	00	100	04	111	00
165654	:	4	011	00	00	100
165659	:	04	71	16	00	00
165664	:	70	15	00	100	04
165669	:	71	16	004	40	00
165674	:	004	100	041	001	00
165679	:	000	005	000	04	00
165684	:	000	000	001	041	001

16514 à 16686
Partie ÉMISSION

17210	:	00	0	00	100	00
17215	:	04	0	40	040	00
17220	:	05	04	4	00	05
17225	:	04	0	40	04	10
17230	:	05	04	4	40	04
17235	:	1	04	0	0	0
17240	:	04	1	04	0	0
17245	:	0	04	1	00	00
17250	:	40	10	04	00	00
17255	:	1	41	1	00	04
17260	:	0040	07	00	00	00
17265	:	0000	07	00	00	00
17270	:	001	07	00	00	00
17275	:	0	00	00	00	00
17280	:	1	0	04	00	00
17285	:	00	00	00	00	00
17290	:	00	10	00	00	00
17295	:	100	70	00	00	00
17300	:	0	0	0	0	0
17305	:	0	0	0	0	0
17310	:	0	0	0	0	0
17315	:	0	0	0	0	0

17210 à 17297
Partie INITIALISATIONS

17360	42	12	64	229	35
17365	1	41	1	0	34
17370	14	64	33	57	64
17375	54	33	35	54	12
17380	225	220	1	33	0
17385	0	200	1	74	1
17390	037	176	201	0	0

17360 à 17392
Partie SCROLLING
PARTIEL

17480	0	33	154	50	126
17485	254	0	40	7	213
17490	195	201	68	209	24
17495	240	33	152	50	126
17500	230	31	33	154	50
17505	54	0	0	0	54
17510	1	30	60	131	111
17515	126	254	255	32	4
17520	30	32	24	212	254
17525	254	32	4	30	0
17530	24	204	254	64	56
17535	2	22	0	50	124
17540	64	205	142	60	244
17545	100	0	0	0	0
17550	50	250	57	50	57
17555	64	50	251	57	50
17560	50	64	50	124	64
17565	42	243	57	34	14
17570	64	215	50	57	64
17575	254	1	32	5	213
17580	205	200	57	209	33
17585	57	64	126	50	250
17590	67	35	126	50	251
17595	67	42	14	64	34
17600	240	67	261	0	0
17605	0	0	1	0	0
17610	205	107	2	60	77
17615	121	60	32	0	62
17620	0	50	199	60	195
17625	05	60	50	199	60
17630	167	194	05	60	62
17635	1	50	199	60	60
17640	0	0	0	0	0
17645	205	109	7	126	50
17650	123	64	195	50	60
17655	42	252	67	209	1
17660	60	0	0	237	75
17665	16	64	167	237	60
17670	225	40	10	34	14
17675	64	237	75	254	67
17680	237	67	57	64	50
17685	120	64	215	62	3
17690	215	42	14	64	237
17695	75	57	64	121	254
17700	32	32	7	40	40
17705	14	1	4	24	2
17710	43	12	34	252	67
17715	237	67	254	67	195
17720	05	60	0	167	254
17725	64	40	2	24	101

17480 à 17960
Corps du programme

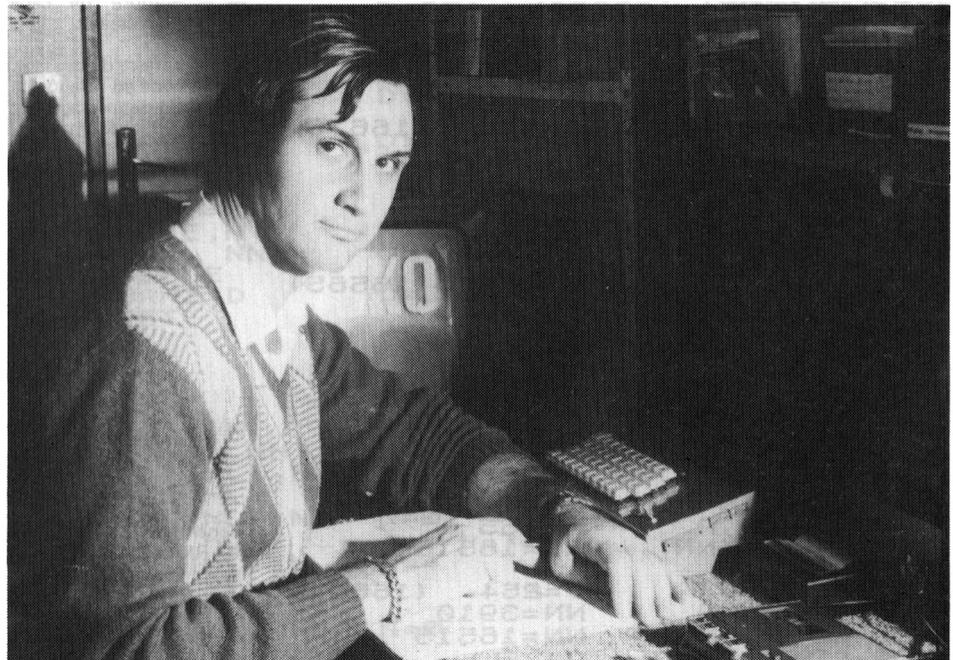
17730	58	103	64	254	114
17735	104	102	69	404	202
17740	67	207	75	204	07
17745	101	054	02	004	14
17750	43	43	34	14	64
17755	14	1	4	007	67
17760	57	64	24	22	40
17765	34	14	64	10	207
17770	67	57	64	40	14
17775	64	207	75	57	64
17780	34	202	67	207	67
17785	254	67	62	0	215
17790	202	0	215	105	05
17795	68	054	217	005	05
17800	2005	105	0	105	005
17805	2004	2054	204	405	005
17810	2004	202	304	0	005
17815	2004	70	105	05	0005
17820	2004	219	300	05	000
17825	000	0	24	3	0004
17830	2000	30	4	000	000
17835	2000	210	105	000	000
17840	2009	14	1	40	100
17845	64	17	141	1	205
17850	17	60	055	50	01
17855	10	6	302	106	107
17860	254	110	302	3	35
17865	24	247	254	0	40
17870	57	254	0	40	30
17875	254	30	56	14	121
17880	04	0	32	6	14
17885	1	60	31	10	19
17890	106	24	12	121	254
17895	0	40	5	14	0
17900	202	27	16	10	126
17905	209	38	70	100	40
17910	111	106	205	050	10
17915	19	246	106	60	20
17920	10	10	60	0	10
17925	10	24	104	00	205
17930	10	201	120	100	100
17935	100	100	100	100	100
17940	40	10	64	17	74
17945	1	35	07	102	170
17950	200	126	254	110	40
17955	245	60	0	110	24
17960	241	240	100	100	100

18480	4	0	0	0	0
18485	0	0	0	0	0
18490	0	5	0	055	14
18495	205	15	10	0	0
18500	30	17	3	0	20
18505	00	10	20	20	200
18510	10	1	100	10	201
18515	7	6	24	3	205
18520	14	0	1	10	200
18525	200	0	11	10	100
18530	20	10	24	20	200
18535	10	50	10	30	00
18540	19	29	21	17	0

18480 à 18544
Table transcodage
émission

17400	0	40	0	00	0
17410	00	46	00	0	41
17418	00	47	01	40	40
17420	40	07	00	40	00
17430	40	00	00	04	00
17430	00	44	00	00	41
17430	00	25	0	01	00
17440	00	0	11	00	00
17440	0	00	00	0	00
17450	40	14	16	00	00
17450	17	00	40	04	00
17460	00	07	10	0	00
17460	07	04	00	054	00

17408 à 17472
Table transcodage
réception



F6GKQ—DENIS A SA STATION

ÉMISSION RTTY

16516 -LD A,N	N=4	Génération du signal MARK
16520 -OUT N,A	N=255	
16523 -LD A,(NN)	NN=16515	
16525 -LD B,A		
16526 -DJNZ D	D=254 (16526)	
16528 -CALL NN	NN=3910	
16531 -LD A,(NN)	NN=16515	
16534 -LD B,A		
16535 -DJNZ D	D=254 (16535)	
16537 -IN A,N	N=254	
16539 -ARR		BC = nombre de caractères du message
16540 -ARR		
16541 -JRC D	D=231 (16518)	
16543 -LD BC,NN	NN=0	
16546 -CALL NN	NN=3910	
16549 -RET NC		
16550 -NOP		
16551 -LD HL,NN	NN=16700	
16554 -ADD HL,BC		
16555 -LD A,(HL)		
16556 -CP N	N=255	HL = adresse du message trans- codé en «Baudot» 255 = fin de message Appel S/P génération Space (Start) D = compteur de moments C = 1 signal MARK C = 0 signal SPACE
16558 -JRNZ D	D=118 (16678)	
16560 -CALL NN	NN=16647	
16563 -LD D,N	N=5	
16565 -ARR		
16566 -CALL C,NN	NN=16616	
16569 -CALL NC,NN	NN=16647	
16572 -DEC D		
16573 -JRNZ D	D=246 (16565)	
16575 -PUSH BC		
16576 -NOP		Génération de 1,5 moment de MARK = STOP
16577 -NOP		
16578 -NOP		
16579 -LD A,(NN)	NN=16514	
16582 -LD L,A		
16583 -ARR		
16584 -ADD A,L		
16585 -LD L,A		
16586 -LD A,N	N=4	
16588 -OUT N,A	N=255	
16590 -LD A,(NN)	NN=16515	
16593 -LD B,A		
16594 -DJNZ D	D=254 (16594)	
16596 -CALL NN	NN=3910	
16599 -LD A,(NN)	NN=16515	
16602 -LD B,A		
16603 -DJNZ D	D=254 (16603)	
16605 -DEC L		
16606 -JRNZ D	D=234 (16586)	
16608 -POP BC		
16609 -INC BC		
16610 -JR D	D=190 (16546)	
16612 -LD L,N	N=255	

```

16614-RET
16615-RLCA
16616-PUSH AF
16617-PUSH BC
16618-LD A, (NN)      NN=16514
16621-LD L, A
16622-LD A, N          N=4
16624-OUT N, A         N=255
16626-LD A, (NN)      NN=16515
16629-LD B, A
16630-DJNZ D           D=254 (16630)
16632-CALL NN         NN=3910
16635-LD A, (NN)      NN=16515
16638-LD B, A
16639-DJNZ D           D=254 (16639)
16641-DEC L
16642-JRNZ D           D=234 (16622)
16644-POP BC
16645-POP AF
16646-RET
16647-PUSH AF
16648-PUSH BC
16649-LD A, (NN)      NN=16516
16652-LD L, A
16653-LD A, N          N=4
16655-OUT N, A         N=255
16657-LD A, (NN)      NN=16517
16660-LD B, A
16661-DJNZ D           D=254 (16661)
16663-CALL NN         NN=3910
16666-LD A, (NN)      NN=16517
16669-LD B, A
16670-DJNZ D           D=254 (16670)
16672-DEC L
16673-JRNZ D           D=234 (16653)
16675-POP BC
16676-POP AF
16677-RET
16678-LD E, N          N=255
16680-CALL NN         NN=16616
16683-DEC E
16684-JRNZ D           D=250 (16680)
16686-RET

```

Sous-programme de génération
du signal MARK

Sous-programme de génération
du signal SPACE

MARK de fin de transmission

```

17210-LD E, N          N=0
17212-LD HL, NN        NN=15001
17215-LD (HL), N       N=0
17217-DEC HL
17218-LD (HL), N       N=0
17220-INC HL
17221-LD (HL), N       N=4
17223-INC HL
17224-INC HL
17225-LD (HL), N       N=0
17227-DEC HL
17228-LD (HL), N       N=13
17230-INC HL
17231-LD (HL), N       N=4
17233-DEC HL
17234-LD (HL), N       N=1
17236-LD (HL), N       N=5
17238-NOP

```

RÉCEPTION RTTY

Initialisations PIA

Accès à DDRA
PORT A en entrée

Accès à ORA

Accès à DDRB

Configuration des lignes
PORT

PB0 = 1 puis PB0 = 1 PB2 = 1

```

17239-NOP
17240-LD (HL),N N=1
17242-LD (HL),N N=9
17244-NOP
17245-NOP
17246-LD (HL),N N=1
17248-NOP
17249-NOP
17250-LD HL,(NN) NN=16395
17253-INC HL
17254-PUSH HL
17255-LD BC,NN NN=297
17258-ADD HL,BC
17259-LD (NN),HL NN=17400
17262-LD A,N N=33
17264-LD (NN),A NN=17402
17267-LD A,N N=15
17269-LD (NN),A NN=17403
17272-NOP
17273-NOP
17274-NOP
17275-NOP
17276-NOP
17277-POP HL
17278-LD BC,NN NN=396
17281-ADD HL,BC
17282-LD (NN),HL NN=17404
17285-LD A,N N=33
17287-LD (NN),A NN=17405
17290-LD A,N N=12
17292-LD (NN),A NN=17407
17295-JP NN NN=17480
17298-NOP
17299-NOP
17300-NOP
17301-NOP
17302-NOP
17303-NOP

```

```

17350-LD HL,(NN) NN=16395
17363-PUSH HL
17364-INC HL
17365-LD BC,NN NN=297
17368-ADD HL,BC
17369-LD (NN),HL NN=16395
17372-LD HL,NN NN=16441
17375-LD (HL),N N=33
17377-INC HL
17378-LD (HL),N N=12
17380-POP HL
17381-PUSH HL
17382-LD BC,NN NN=33
17385-ADD HL,BC
17386-POP DE
17387-LD BC,NN NN=330
17390-LDIR
17392-RET
17393-NOP
17394-NOP
17395-NOP
17396-NOP

```

] petite tempo
PBO = 1 puis PBO = 1,
PB2 = 0, PB3 = 1
] tempo

Initialisations partage ÉCRAN

Pointe et sauvegarde début écran

Pointe la 9ème ligne et sauve en 17400 (zone décodage) (DFCC - DEC)
Prépare les nième lignes et colonnes décodage
17402 = C - DEC
17403 = L - DEC

Prépare les positions curseur écriture

17404 = DFCC - ECR
17406 = C - ECR (colonne)
17407 = L - ECR (ligne)
saute en 17480, corps du programme

ROUTINE DE SCROLLING PARTIEL

Pointe D-FILE et sauvegarde
Pointe DFCC et calcule le nombre de lignes à déplacer (ici $9 \times 33 = 297$ octets)
calcule la nouvelle position d'écriture et renseigne les pointeurs colonne et ligne (S - POSN)

Pointe le début de la 2ème ligne (D-FILE + 33)

Effectue le déplacement des 330 octets

```

17480-NOP
17481-LD HL,NN
17484-LD A,(HL)
17485-CP N
17487-JRZ D
17489-PUSH DE
17490-JP NN
17493-POP DE
17494-JR D
17496-LD HL,NN
17499-LD A,(HL)
17500-AND N
17502-LD HL,NN
17505-LD (HL),N
17507-NOP
17508-NOP
17509-LD (HL),N
17511-LD H,N
17513-ADD A,E
17514-LD L,A
17515-LD A,(HL)
17516-CP N
17518-JRNZ D
17520-LD E,N
17522-JR D
17524-CP N
17526-JRNZ D
17528-LD E,N
17530-JR D
17532-CP N
17534-JRC D
17536-LD A,N
17538-LD (NN),A
17541-CALL NN
17544-JR D
17546-NOP
17547-NOP
17548-NOP
17549-NOP
17550-LD A,(NN)
17553-LD (NN),A
17556-LD A,(NN)
17559-LD (NN),A
17562-LD A,(NN)
17565-LD HL,(NN)
17568-LD (NN),HL
17571-RST 16
17572-LD A,(NN)
17575-CP N
17577-JRNZ D
17579-PUSH DE
17580-CALL NN
17583-POP DE
17584-LD HL,NN
17587-LD A,(HL)
17588-LD (NN),A
17591-INC HL
17592-LD A,(HL)
17593-LD (NN),A
17596-LD HL,(NN)
17599-LD (NN),HL
17602-RET
17603-NOP
17604-NOP
17605-NOP
17606-NOP

```

```

NN=15002
N=3
D=7 (17496)
NN=17509
D=240 (17480)
NN=15000
N=31
NN=15002
N=0
N=1
N=68
N=255
D=4 (17524)
N=32
D=212 (17480)
N=254
D=4 (17532)
N=0
D=204 (17480)
N=64
D=2 (17538)
N=0
NN=16508
NN=17550
D=190 (17480)

```

PROGRAMME PRINCIPAL

Teste si le registre UART est plein. Si oui, saute à la lecture du caractère ; si non, saute à la scrutation clavier
Recommence au retour...

Lit le caractère présent dans l'UART et envoie une impulsion d'effacement pour libérer le registre UART

Transcodage Baudot/Sinclair

Teste si préfixe chiffres/lettres

Passage en chiffres ou lettres si ce n'est déjà fait

Effectue un test de validité (< 64) sur le code du caractère à afficher
Mémorise en 16508 le caractère et appelle la routine d'affichage. Au retour, on revient sur la lecture UART.

Routine AFFIDEC (affichage caract. décodé)

Restaure les pointeurs d'écriture «décodage»

Affiche le caractère mémorisé

Sauvegarde le contexte et effectue un scrolling si on est sur la dernière colonne
(17360 = scrolling partiel)

Calcule et mémorise la prochaine position d'écriture «décodage»

des NOP de 17603 à 17609

```

NN=17402
NN=16441
NN=17403
NN=16442
NN=16508
NN=17400
NN=16398
NN=16441
N=1
D=5 (17584)
NN=17360
NN=16441
NN=17402
NN=17403
NN=16398
NN=17400

```

```

17610-CALL NN
17613-LD B,H
17614-LD C,L
17615-LD A,C
17616-INC A
17617-JRNZ D
17619-LD A,N
17621-LD (NN),A
17624-JP NN
17627-LD A,(NN)
17630-AND A
17631-JPNZ NN
17634-LD A,N
17636-LD (NN),A
17639-NOP
17640-NOP

```

```

NN=599
D=8 (17627)
N=0
NN=17607
NN=17493
NN=17607
NN=17493
N=1
NN=17607

```

```

17641-NOP
17642-NOP
17643-NOP
17644-NOP
17645-CALL NN
17648-LD A,(HL)
17649-LD (NN),A
17652-JP NN
17655-LD HL,(NN)
17658-PUSH HL
17659-LD BC,NN
17662-ADD HL,BC
17663-LD BC,(NN)
17667-AND A
17668-SBC HL,BC
17670-POP HL
17671-JRNC D
17673-LD (NN),HL
17676-LD BC,(NN)
17680-LD (NN),BC
17684-LD A,(NN)
17687-RST 16
17688-LD A,N
17690-RST 16
17691-LD HL,(NN)
17694-LD BC,(NN)
17698-LD A,C
17699-CP N
17701-JRNZ D
17703-DEC HL
17704-DEC HL
17705-LD C,N
17707-INC B
17708-JR D
17710-DEC HL
17711-INC C
17712-LD (NN),HL
17715-LD (NN),BC
17719-JP NN
17722-NOP
17723-AND A
17724-CP N
17726-JRNC D
17728-JR D
17730-LD A,(NN)
17733-CP N

```

```

NN=1981
NN=16507
NN=17722
NN=17404
NN=66
NN=16400
D=16 (17691)
NN=16396
NN=17406
NN=16441
NN=16507
N=3
NN=16396
NN=16441
N=32
D=7 (17710)
N=1
D=2 (17712)
NN=17404
NN=17406
NN=17493
N=64
D=2 (17730)
D=161 (17655)
NN=16507
N=114

```

Routine d'exploration du clavier

Explore le clavier et détecte si une touche est enfoncée. Si oui, décodage de la touche, si non retour à la lecture UART. Effectue un test anti-rebond et revient à la lecture UART tant que la même touche est maintenue. 17607 sert de variable «anti-rebond»

*(suite de
la lecture CLAVIER)*

On appelle la routine de décodage (ROM). Le code de la touche est préservé en 16507 et on saute au «traitement touche»

AFFIECR Routine affichage caractère

Restaure les pointeurs d'écriture et teste si ce n'est pas la dernière position d'écran

Affiche le caractère saisi au clavier et affiche à la position suivante le curseur d'écriture

Sauvegarde, après calcul, la prochaine position d'écriture

Puis effectue un retour

à la lecture de

l'UART

ROUTINE DE TRAITEMENT

Teste si c'est un caractère (< 64) ou une fonction. Si caractère, saute à AFFIECR. Teste si ← retour arrière effacement

17735-JPNZ NN	NN=17795
17738-LD HL,(NN)	NN=17404
17741-LD BC,(NN)	NN=17405
17745-LD A,C	
17746-CP N	N=32
17748-JRNZ D	D=14 (17764)
17750-DEC HL	
17751-DEC HL	
17752-LD (NN),HL	NN=16395
17755-LD C,N	N=1
17757-INC B	
17758-LD (NN),BC	NN=15441
17762-JR D	D=9 (17773)
17764-DEC HL	
17765-LD (NN),HL	NN=16395
17768-INC C	
17769-LD (NN),BC	NN=16441
17773-LD HL,(NN)	NN=16395
17776-LD BC,(NN)	NN=15441
17780-LD (NN),HL	NN=17404
17783-LD (NN),BC	NN=17405
17787-LD A,N	N=3
17789-RST 16	
17790-LD A,N	N=0
17792-RST 16	
17793-JP NN	NN=17493
17796-CP N	N=217
17798-JRNZ D	D=6 (17805)
17800-CALL NN	NN=2153
17803-JP NN	NN=17493
17806-CP N	N=224
17808-JRZ D	D=30 (17840)
17810-CP N	N=221
17812-JRNZ D	D=0 (17820)
17814-CALL NN	NN=17940
17817-JP NN	NN=17493
17820-CP N	N=219
17822-JRNZ D	D=5 (17829)
17824-POP DE	
17825-LD E,N	N=0
17827-JR D	D=7 (17836)
17829-CP N	N=220
17831-JRNZ D	D=4 (17837)
17833-POP DE	
17834-LD E,N	N=32
17836-PUSH DE	
17837-JP NN	NN=17493
17840-POP DE	
17841-LD C,N	N=1
17843-LD HL,(NN)	NN=16395
17846-LD DE,NN	NN=397
17849-ADD HL,DE	
17850-LD DE,NN	NN=16700
17853-LD A,N	N=31
17855-LD (DE),A	
17856-LD B,N	N=0
17858-LD A,(HL)	
17859-AND A	
17860-CP N	N=118
17862-JRNZ D	D=3 (17867)
17864-INC HL	
17865-JR D	D=247 (17858)
17867-CP N	N=3

Gestion de la position du curseur en marche arrière pour effacement

Calcule la prochaine position Y affichera le curseur écriture Efface la position présente Sauvegarde le contexte «écriture» Effectue un retour à la lecture UART

ROUTINE TEST FONCTIONS

Si shift W, on effectue un COPY

Si shift E, on saute à PREPAR EMISSION
Si shift T, on appelle 17940, effacement partiel écran

Si shift R, on force en mode lettres

Si shift Y, on force en mode chiffres

Si touche interdite, retour UART

PREPAR EMISSION

Pointe le début de l'écran et le début de la zone qui contiendra le message transcodé (16700)

Lit la position écran pointée

Teste si 118 et incrémente si oui teste si curseur fin de message

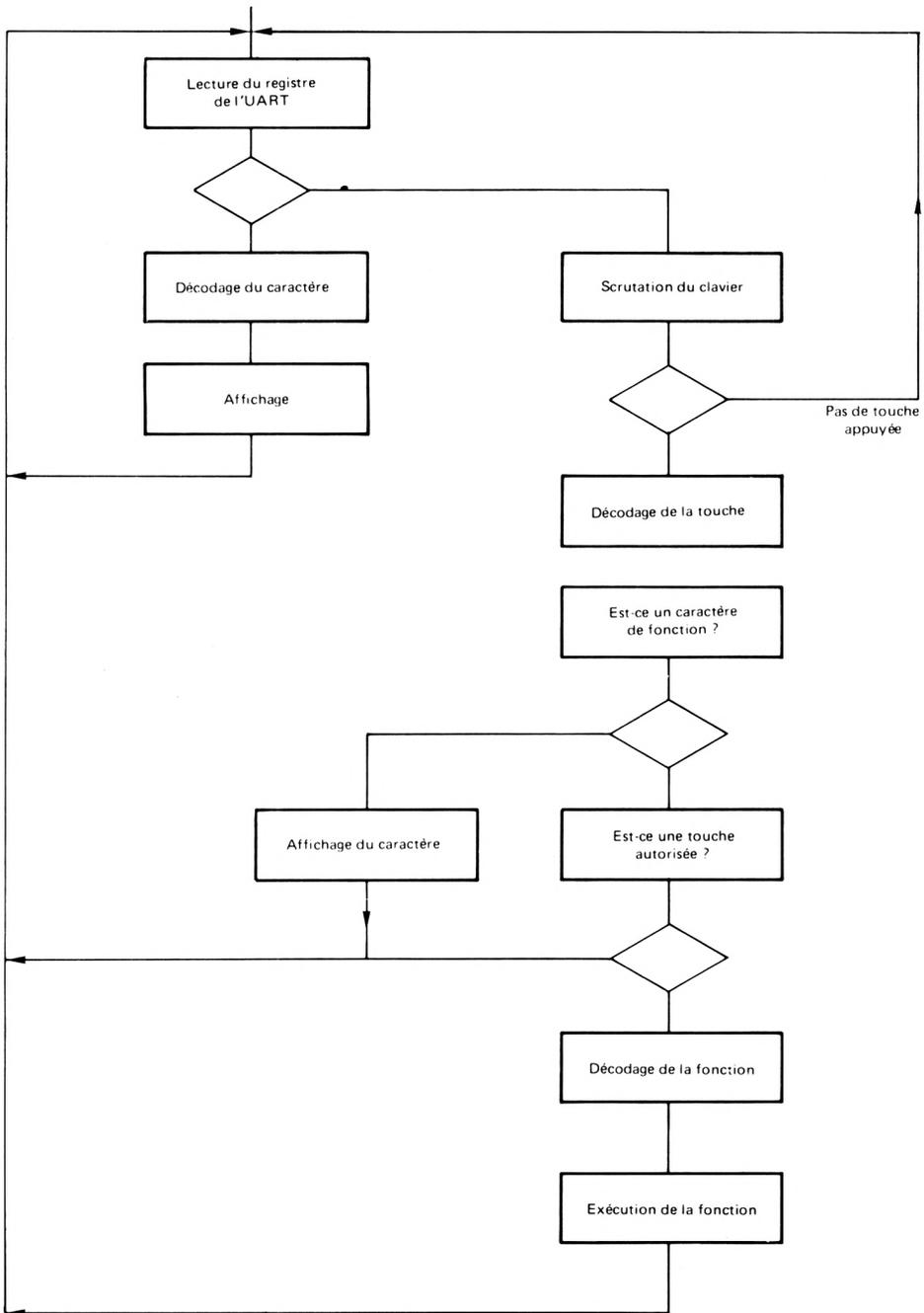
17869-JRZ D	D=57 (17928)	
17871-CP N	N=0	Teste si espace
17873-JRZ D	D=90 (17905)	
17875-CP N	N=30	
17877-JRC D	D=14 (17893)	Teste si lettre ou chiffre
17879-LD A,C		
17880-CP N	N=0	
17882-JRNZ D	D=6 (17890)	
17884-LD C,N	N=1	
17886-LD A,N	N=31	Charge le préfixe lettres si on était en mode chiffres
17888-LD (DE) ,A		
17889-INC DE		
17890-LD A, (HL)		
17891-JR D	D=12 (17905)	
17893-LD A,C		
17894-CP N	N=0	
17896-JRZ D	D=6 (17904)	
17898-LD C,N	N=0	
17900-LD A,N	N=27	
17902-LD (DE) ,A		Charge le préfixe chiffres si on était en mode lettres
17903-INC DE		
17904-LD A, (HL)		
17905-PUSH HL		
17906-LD H,N	N=72	
17908-ADD A,N	N=40	Pointe la table de transcodage
17910-LD L,A		
17911-LD A, (HL)		
17912-POP HL		
17913-INC HL		
17914-LD (DE) ,A		
17915-INC DE		
17916-JR D	D=196 (17853)	
17916-LD A,N	N=2	Emet les avances papier et retours chariot si besoin est (dans ce cas, en 17916 on a DJNZ au lieu de JR)
17920-LD (DE) ,A		
17921-INC DE		
17922-LD A,N	N=8	
17924-LD (DE) ,A		
17925-INC DE		
17926-JR D	D=164 (17856)	
17928-LD A,N	N=255	Place l'indicateur fin de message
17930-LD (DE) ,A		
17931-RET		

17940-LD HL, (NN)	NN=16396	
17943-LD DE, NN	NN=330	
17946-INC HL		
17947-DEC DE		
17948-LD A,D		
17949-OR E		
17950-RET Z		
17951-LD A, (HL)		
17952-CP N	N=116	
17954-JRZ D	D=246 (17946)	
17956-LD A,N	N=0	
17958-LD (HL) ,A		
17959-JR D	D=241 (17946)	

EFFACEMENT PARTIEL DE L'ÉCRAN

Pointe le début du fichier DE contient le nombre de caractères à effacer

Saute les «new-line»... met 0 à l'emplacement pointé



ORGANIGRAMME SIMPLIFIÉ
DU PROGRAMME RTTY

PROGRAMME "FICHIER 2"

Programme très simple présentant une des différentes manières d'aborder un problème de tri. Dans ce programme nous procéderons à un tri alphanumérique. C'est un type de tri couramment utilisé, par exemple, pour ranger des cartes QSL avant de les expédier au bureau ...

Le tri alphanumérique sur le ZX est une chose réellement aisée car la fonction de comparaison l'utilise lorsqu'on traite des chaînes de caractères.

Ainsi on trouve «ABC» < «ACB» avec le BASIC du ZX. Ceci est bien pratique !

Dans le programme, on dimensionne deux tableaux en fonction du nombre d'éléments à trier. C'est le rôle des lignes 350 et 400. La première liste (L\$) contiendra les éléments introduits dans le désordre ; la deuxième (O\$) sera la liste ordonnée.

La longueur maximum des éléments à entrer est de 8 caractères, ce qui évidemment est modifiable.

Si on donne un nombre d'éléments à classer, supérieur au nombre réel, on peut quitter la phase d'introduction des éléments en tapant . (point). Le rôle de la ligne 660 est de reconnaître ceci.

Lors du tri, on considère arbitrairement (il faut bien commencer par quelque chose) le premier élément comme étant le plus «faible», et on le met dans F\$. Les comparaisons et le contenu de F\$ sont modifiés quand on rencontre un élément de rang inférieur (ligne 1050).

L'élément qui vient d'être rangé dans la liste ordonnée est supprimé de la liste de départ et remplacé par un «marqueur» (ici code 128 du carré noir). Lorsqu'on rencontre ce marqueur lors du tri, on passe à l'élément suivant.

Pour éditer la liste classée, on a choisi d'effectuer un scrolling de l'écran, qui fera défiler la liste des éléments classés par ordre alphanumérique.

Une pression sur une touche permettra d'arrêter le scrolling et une nouvelle pression le redéarrera. C'est le rôle de la ligne 1750.

On pourra recommencer un autre classement.

Ce programme n'étant présenté qu'à titre d'exemple pour démontrer la facilité d'utilisation des comparaisons de chaînes pour le tri alphanumérique, il est possible de modifier la fin pour proposer un «menu» d'options permettant : la sauvegarde de la liste classée, la copie sur imprimante, le retour au tri, etc...

```
10 REM "FICHIER2"
15 REM
20 REM F6GKQ...1981
25 REM
30 FAST
40 LET N=0
100 LET S=0
150 PRINT "PROGRAMME DE TRI ALP
ANUMERIQUE"
200 PRINT "COMBIEN D""ELEMENTS
A CLASSER ?"
```

```

250 INPUT NE
350 DIM L$(NE,8)
400 DIM O$(NE,8)
450 CLS
500 PRINT AT 19,0;"INTRODUISEZ
510 ELEMENTS OU "
540 REM ENTREE DES ELEMENTS
550 FOR I=1 TO NE
600 SCROLL
650 INPUT L$(I)
660 IF CODE L$(I) <> 27 THEN GOTO
700
670 LET NE=I-1
680 GOTO 500
700 PRINT L$(I)
750 NEXT I
800 CLS
840 REM TRI
850 FOR I=1 TO NE
900 IF CODE L$(I)=126 THEN GOTO
1550
950 LET F#=L$(I)
1000 FOR J=1 TO NE
1050 IF L$(J) < F# THEN LET F#=L$(
J)
1100 NEXT J
1150 LET N=N+1
1200 LET O$(N)=F#
1250 IF N=NE THEN GOTO 1590
1290 REM ELIMINE ELEMENT CLASSE
1300 FOR K=1 TO NE
1350 IF L$(K) < F# THEN GOTO 1500
1400 LET L$(K)=" "
1450 GOTO 1550
1500 NEXT K
1550 NEXT I
1560 IF N < NE THEN GOTO 850
1590 CLS
1600 REM EDITION DU FICHIER
1610 SLOW
1650 FOR I=1 TO N
1700 SCROLL
1750 IF INKEY$ <> "" THEN PAUSE 4E
4
1760 POKE 16437,255
1800 PRINT I;TAB 4;O$(I)
1850 NEXT I
1860 SCROLL
1900 PRINT AT 21,0;"NEW-LINE FOR
REFERENCE"
1950 PAUSE 4E4
1960 POKE 16437,255
1970 CLS
1980 GOTO 50
2250 SAVE "FICHIER2"
2300 GOTO 50

```

INTRODUISEZ LES ELEMENTS OU

FRBR/MM
FR1CR
SB4MY
WB4RTY
F6GKQ
G4NNG
DU9ZU
F1EZH
PY2UE
9H100

PRESENTATION DE L'ÉCRAN AVANT CLASSEMENT

1 SB4MY
2 9H100
3 DU9ZU
4 FR1CR
5 F1EZH
6 FRBR/MM
7 F6GKQ
8 G4NNG
9 PY2UE
10 WB4RTY

LINE POUR RECOMMENCER

APRES LE CLASSEMENT ALPHANUMÉRIQUE

TROISIEME PARTIE

AMELIORATION DES ENTREES-SORTIES

- PIA**
- UTILISATION DU PORT D'ENTREE ZX**
- UNE CARTE 8 ENTREES - 8 SORTIES**
- UN PROGRAMMATEUR D'EPROM**
- UN CIRCUIT BIP SONORE**

MODIFICATION ET AMELIORATION DU GRAPHISME

- GRAPHISME SUR IMPRIMANTE**
 - GENERATEUR DE CARACTERES PROGRAMMABLES**
- ## **CONCLUSIONS**

AMELIORATION DES ENTREES-SORTIES

Le manque de possibilités d'entrées-sorties sur le ZX peut facilement être arrangé. Il existe pour cela plusieurs possibilités, allant des simples latches aux circuits spécialisés.

Nous allons examiner l'utilisation d'un des plus adaptés : le P.I.A. Pour des raisons de prix et de facilité de programmation, même en BASIC, nous avons choisi le 6821 plutôt que l'outil spécialisé de la famille Z 80 (le PIO, testé néanmoins avec succès sur une autre carte programmable en langage machine).

Le PIA (peripheral interface adapter) est un outil très puissant permettant de programmer deux groupes de 8 lignes d'entrées-sorties (on dit aussi des PORTS) dont on peut, pour chacune, définir le sens.

Les applications sont donc multiples et sortent du cadre de cet ouvrage. Nous nous contentons de vous en fournir un schéma (avec une variante selon que votre ZX ait été, ou non, mis dans un boîtier plus grand) et comme exemple d'application, nous vous renvoyons au programme RTTY où le PIA est utilisé en lecture (caractère en sortie UART) et en écriture (signaux d'initialisation, accusé de réception du caractère).

Même si vous câblez le PIA et l'UART sur une même carte, nous vous conseillons de prévoir un connecteur débrochable facilement entre les lignes PIA et l'UART. Sur ce connecteur seront disponibles toutes les lignes PIA pour d'autres utilisations (Figure 24). Nous présentons, Figure 25, le brochage complet du 6821.

Reportez-vous au schéma du décodeur d'adresse simplifié (Figure 23).

Nous avons prévu deux cas :

- Votre ZX est dans un boîtier plus grand autorisant l'implantation de diverses extensions. Vous ne câblerez pas la diode D1 et le transistor. La sortie Y0 (patte 15 du 74LS138) sera reliée à la résistance R28 (680 Ohms) du ZX après en avoir soulevé la patte gauche, qui sera donc coupée du circuit imprimé. Le ZX est vu côté composants, connecteur de sortie en haut à droite.
ATTENTION : Si vous retirez un jour le 74LS138, il faudra ressouder la résistance R28. C'est la solution la plus fiable.
- Votre ZX est dans son boîtier d'origine et les extensions prévues vont donc se connecter à l'extérieur. Le transistor devra être câblé et sert à forcer la ligne ROMCS. La résistance de 2,2 k Ω sera peut-être à ajuster jusqu'à 10 k Ω , de même qu'il faudra peut-être prévoir une 150 Ω en série dans l'émetteur du transistor, si le ZX ne s'initialise pas bien à chaque fois.

Précisons que, si vous avez monté le 74LS32 comme décrit précédemment dans cet ouvrage, ce transistor devient inutile. Il faut l'éviter dans la mesure du possible car il demande quelques manipulations supplémentaires comme expliqué ci-dessus.

Précisons enfin que, si vous utilisez une extension RAM 64 K, il ne faudra pas valider l'espace 8 K – 16 K (8192 à 16383).

Nous vous engageons à modifier votre ZX avec le montage 74LS32, simple à faire et à câbler comme décodeur d'adresses (surtout s'il est à l'intérieur du ZX), le montage donné précédemment (Figure 9) découpant l'espace 8 K – 16 K en segments de 2 K.

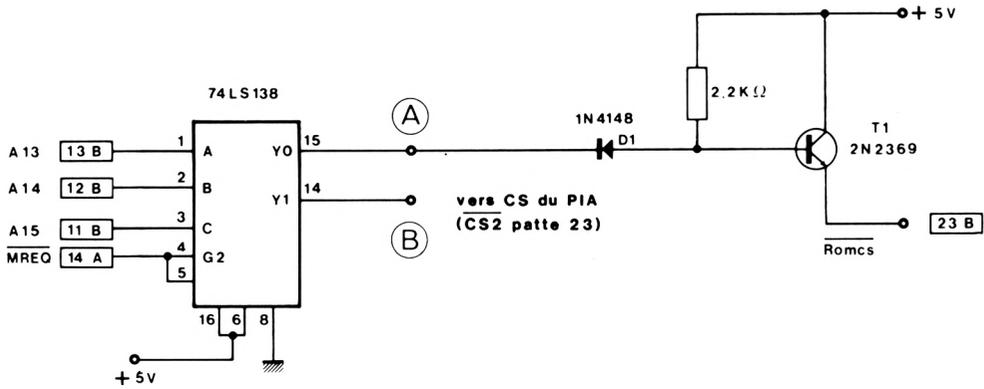


FIGURE 23 : DECODEUR D'ADRESSE SIMPLIFIE.

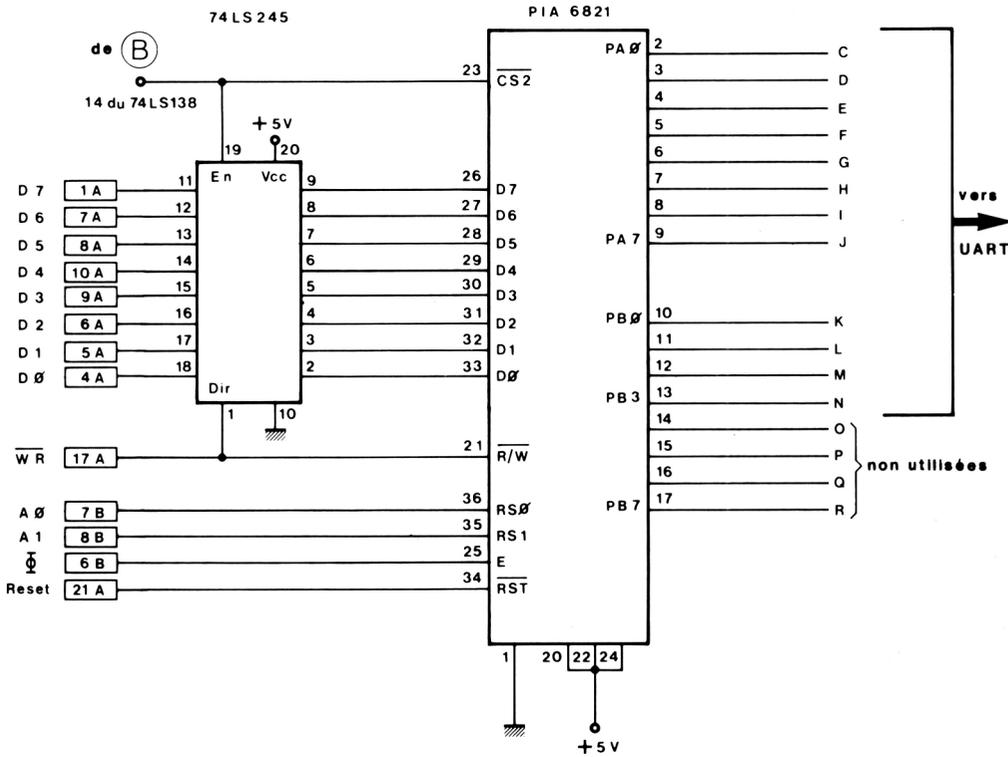


FIGURE 24 : PIA AVEC BUFFER DE BUS

– Relier masse ZX 4B du connecteur à la masse carte PIA.

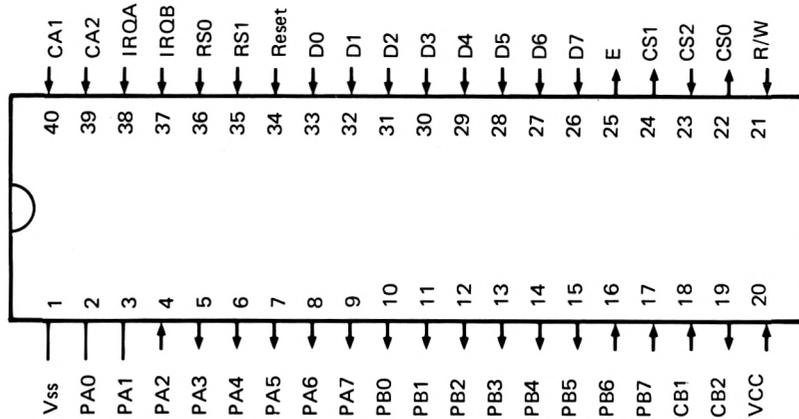
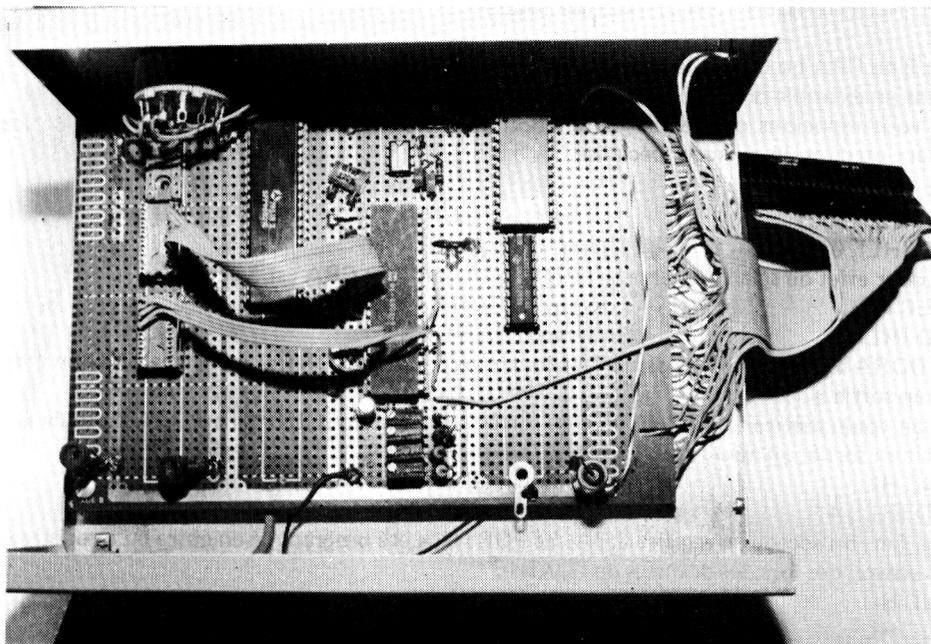


FIGURE 25 : BROCHAGE DU PIA 6821



BOITIER EXTENSION CONTENANT PIA ET UART.

NOTER: — à gauche les supports à wrapper utilisés comme connecteurs
— en bas, au centre, ajustables multi-tours de l'UART.

Programmation du PIA

Avec le décodage d'adresse adopté, le PIA répond à plusieurs adresses. Nous n'en retiendrons que 4 (le PIA occupe 4 emplacements en mémoire).

Nous n'entrerons pas dans le détail de la programmation de ce composant et vous renvoyons aux livres spécialisés, traitant des entrées-sorties. Il faut bien connaître le mécanisme de fonctionnement de ses registres internes.

Si on considère le PORT A, son fonctionnement est déterminé par les registres (même chose pour PORT B : CRB, DDRB, ORB).

CRA	registre de contrôle de fonctions,
DDRA	définition du sens des lignes du PORT,
ORA	registre d'entrée-sortie où sont présentes les données.

Ces registres sont accessibles en fonction de l'état des 2 lignes RS0 - RS1.
Ainsi on trouvera :

DDRA-ORA	adresse 15000
CRA	adresse 15001
DDRB-ORB	adresse 15002
CRB	adresse 15003

Le bit 2 de CRA ou CRB sélectionne ORA ou ORB.

Voyons un exemple pratique, extrait du programme RTTY (17210 à 17249).

Nous vous rappelons que la programmation détaillée du PIA doit être vue dans la notice du constructeur ou dans un ouvrage spécialisé.

LD HL, 15001 on adresse CRA

LD (HL), 0 on met 0 dans CRA

ce qui a pour effet de sélectionner le registre DDRA (bit 2 de CRA est à zéro).

DEC HL on pointe l'adresse 15000

LD (HL), 0 on met 0 dans DDRA

0 dans DDRA a pour effet de positionner toutes les lignes du PORT A en entrée (ligne en entrée = 0 ; ligne en sortie = 1).

Si on avait voulu mettre toutes les lignes en sortie, on aurait écrit 255 dans DDRA. Si seule la ligne PA 3 était en sortie, on aurait mis 8 = 00001000 dans DDRA.

INC HL on pointe l'adresse 15001

LD (HL), 4 on met bit 2 de CRA à 1

et par ce fait, on accède au registre ORA. Le PORT A a été programmé en entrée. Le registre ORA du PIA recevra, dès lors, les données de l'UART.

INC HL

INC HL on pointe sur 15003

LD (HL), 0 met 0 dans CRB

ce qui a pour effet de sélectionner le registre DDRB (bit 2 de CRB à zéro).

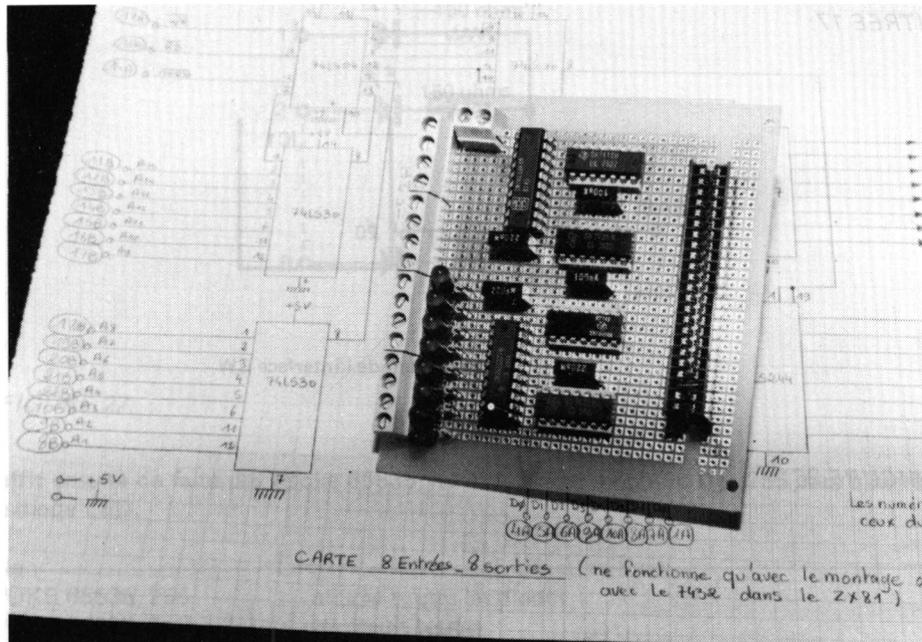
DEC HL on pointe sur 15002

LD (HL), 13 on met 13 = 00001101

13 dans DDRB place PB0, PB2, PB3 en sortie, les autres lignes sont en entrée.

INC HL	on pointe l'adresse 15003
LD (HL), 4	et on met bit 2 de CRB à 1
ce qui permet d'accéder à ORB.	
DEC HL	on pointe 15002
LD (HL), 1	maintient PB0 à 1 et met les autres lignes à zéro
(en fait PB2 et PB3 nous intéressent).	
LD (HL), 5	met PB2 à 1 (PB0 maintenue à 1)
NOP, NOP	petit délai (environ 2,5 µs)
LD (HL), 1	remet PB2 à 0
on a produit une impulsion de 2,5 µs sur PB2 qui active la ligne Master Reset de l'UART.	
LD (HL), 9	met PB3 à 1
NOP, NOP	petit délai (2,5 µs)
LD (HL), 1	remet PB3 à 0
on a produit une impulsion sur PB3, c'est-à-dire sur la ligne CRL de l'UART.	
A partir de cet instant l'UART est initialisé et le PIA va pouvoir lire les informations qu'elle envoie.	

Voici donc un exemple d'application d'une programmation en entrée et sortie du PIA.
 Les sorties du PIA ne doivent pas être chargées inconsidérément. Si vous voulez commander des LED, des relais, etc... passez toujours par un transistor.
 Vous pourrez, par exemple, commander la mise en marche et l'arrêt du magnétophone avant chaque SAVE et LOAD dans un programme...



LA CARTE 8 ENTRES - 8 SORTIES. (voir Page 157)

LE PORT D'ENTREE K7 DU ZX 81

Le ZX 81 a, dans sa version de base, une possibilité de recevoir des données du monde extérieur. Cette possibilité a été exploitée dans le programme CW décrit dans ce livre, mais il est bon de bien connaître son utilisation.

La patte 20 de IC 1 est en réalité l'entrée d'un buffer sur la ligne D7 du bus de données. Ce buffer est commandé, comme un périphérique, avec l'instruction IN. L'adresse de ce buffer est 254 (IN A, 254) et le résultat se trouve donc sur D7 mais inversé, car il s'agit en fait d'un buffer inverseur.

Pour lire ce port, il suffit donc de faire :

IN A, 254 (met dans A le contenu du port 254)
BIT 7,A (lecture entrée K7)

Si le bit 7 est à 0, c'est que la patte 20 de IC 1 est au + 5 V et inversement. Pour protéger le circuit LSI, il est préférable, lors de l'utilisation de ce port, de se brancher, tout comme pour la CW, devant la résistance de 4,7 k.

ENTREE 17:

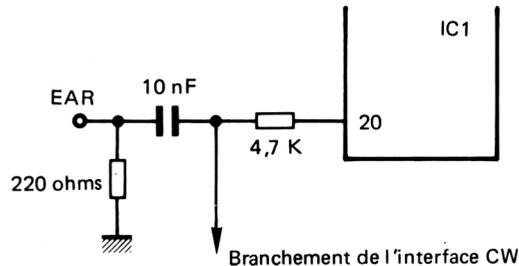
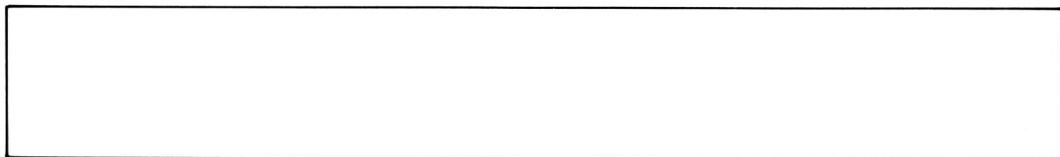


FIGURE 26



LA CARTE 8 ENTREES - 8 SORTIES

Nous avons vu que le PIA est un coupleur d'entrées-sorties fort adapté et très puissant, d'un coût réduit. Son seul inconvénient est peut-être sa complexité de programmation qui découle de la puissance de cet outil.

Il peut s'avérer dans certains cas que l'on n'ait pas besoin de disposer d'autant de possibilités, notamment le mixage d'entrées et sorties sur un même PORT. Dans ce cas, il est possible d'utiliser une carte dont la programmation sera réduite à la plus simple expression. Cette carte offrira 8 lignes d'entrée et 8 lignes de sortie, toutes programmables indépendamment les unes des autres.

En utilisant un décodage d'adresses assez complet, en association avec le montage proposé Figure 8 (74LS32) qui permet de libérer complètement la zone 32768 à 65536, la programmation de la carte pourra se faire sur deux adresses. Celles qui ont été choisies sont 65534 et 65535, ce qui ne devrait poser aucun problème sur des ZX standards, hormis peut-être dans le cas de l'utilisation d'une extension mémoire 64 K Octets, si l'espace supérieur est validé.

La réalisation de la carte, en wrapping ne présente aucune difficulté. Sa connexion sur le bus de sortie du ZX s'effectue grâce à un connecteur femelle ou un ensemble mâle-femelle pour un montage « en coupure » entre le ZX et une autre extension.

Pour vérifier le bon fonctionnement de la carte et en avoir le témoignage permanent, il sera bon de connecter sur les sorties des diodes électroluminescentes comme indiqué ci-dessous.

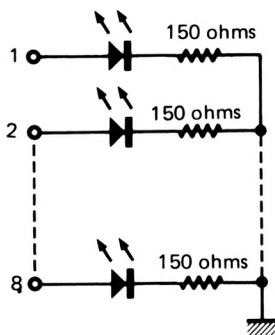


FIGURE 27.

Il suffit ensuite de faire des «poke 65535, N» avec N compris entre 0 et 255 pour allumer telle ou telle diode LED.

Exemple :

POKE 65535, 255

POKE 65535, 0

POKE 65535, 170

allume toutes les diodes

les éteint toutes

allume une diode sur deux.

Les sorties ne supportent pas de grandes charges, il est donc hors de question d'y relier directement, par exemple, un relais. Il sera donc nécessaire, lors de branchements vers le monde extérieur, de prévoir un tampon qui pourra avantageusement être un transistor.

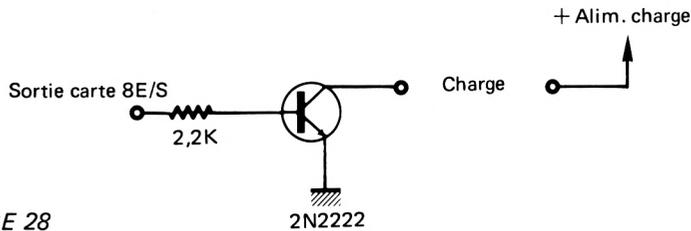


FIGURE 28

ATTENTION : Ne pas se servir de l'alimentation du ZX 81 comme alimentation de la charge, car elle est calculée juste.

Pour les entrées, la mise à la masse ou au + 5 V (via, dans le cas du + 5 V, une résistance de 1 k de préférence) d'une ligne fera passer le bit correspondant à 0 ou 1 suivant le cas. La lecture se fait en faisant, par exemple : PRINT PEEK 65535.

Quand toutes les entrées sont « en l'air », cela équivaut à 1.

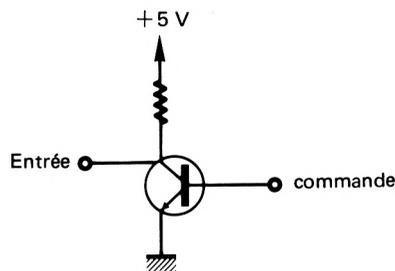
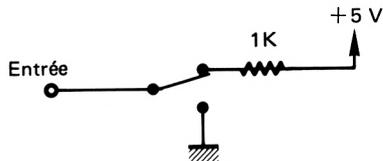


FIGURE 29.

Résumé:

- Adresse sortie: 65535
 entrée: 65535
- sortie donnée: POKE 65535, N ou en L.M.: LD (65535), N
- entrée donnée: PRINT = PEEK 65535
 LET X = PEEK 65535
 etc...

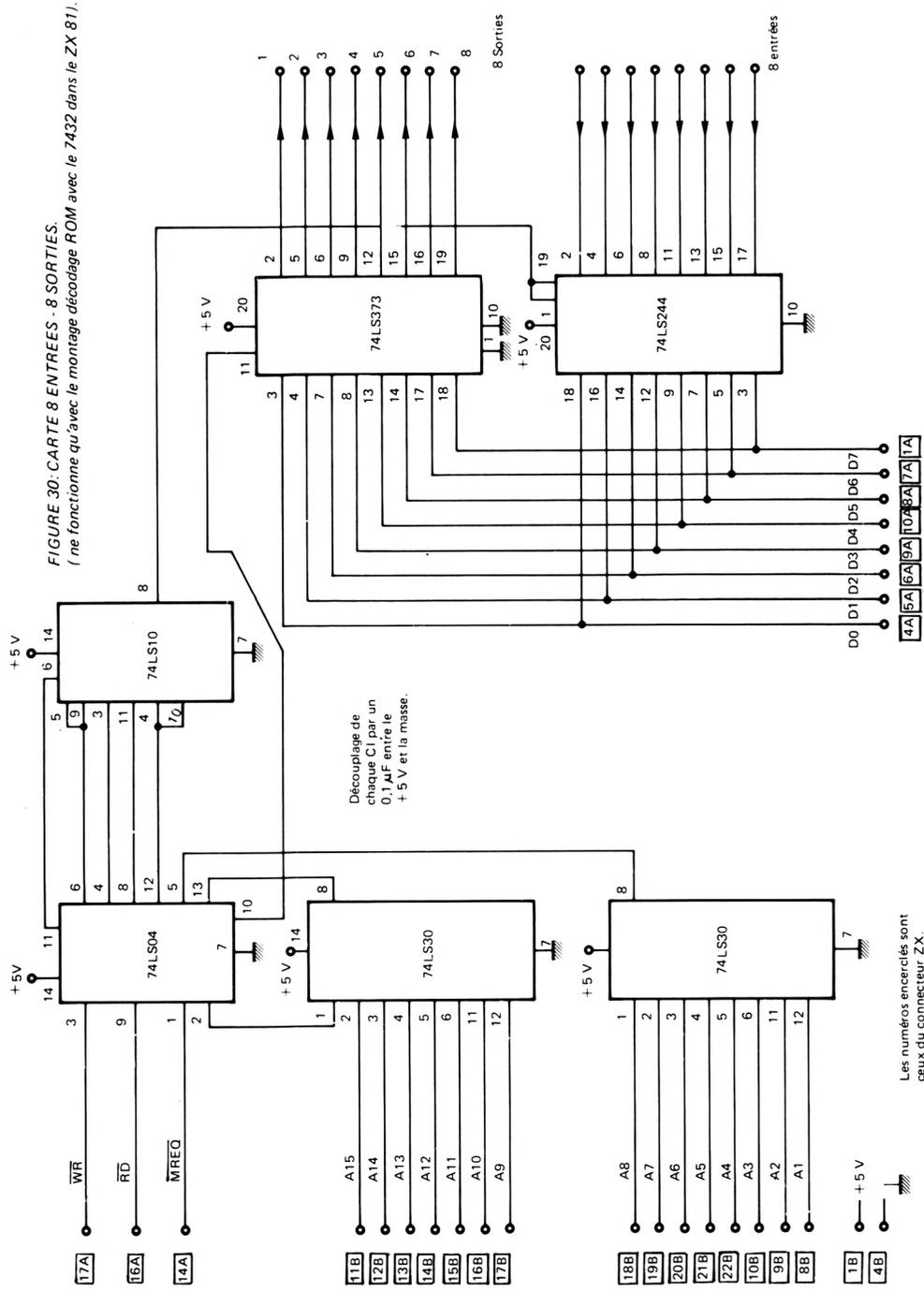


FIGURE 30: CARTE 8 ENTREES - 8 SORTIES.
 (ne fonctionne qu'avec le montage décodage ROM avec le 7432 dans le ZX 81).

Les numéros encadrés sont ceux du connecteur ZX.

PROGRAMMATEUR D'EPROM

Dans la mesure où le système ZX a été équipé d'un circuit d'entrées-sorties, dans notre cas un PIA, il est facile de faire exécuter à l'ensemble des tâches complexes, telles que la programmation d'EPROM.

Petits rappels :

Une EPROM (Erasable programmable read only memory) est un circuit mémoire « morte » présentant la particularité d'être programmable.

Pour ce faire, il suffit d'effacer le contenu précédent de la mémoire, en l'exposant à une source de lumière ultra-violette spécifique. Le constructeur préconise une longueur d'ondes $\lambda = 2537 \text{ \AA}$. La durée de l'exposition varie de 20 minutes à 1 heure selon l'intensité de la source UV, disposée à environ 2,5 cm de la fenêtre d'effacement.

Après un effacement correct, tous les bits de la mémoire sont positionnés à 1. Si une lecture de la mémoire affirme le contraire, l'effacement n'est pas correct. Un bit resté à zéro ne pourra pas être programmé.

Pour la programmation d'une mémoire on aura besoin d'une source « haute tension » à 25 V, 30 mA.

La programmation d'une telle mémoire est simple. On présente à la mémoire l'adresse, la donnée correspondante et une impulsion sur une broche spécialisée qui permet la prise en compte de l'information.

Ceci est reproduit cycliquement jusqu'au dernier mot à programmer.

Comme il y a quelques impératifs à respecter, la gestion par microprocesseur se justifie pleinement (les adresses et les données doivent être stabilisées quand on envoie l'impulsion de programmation dont la durée est assez stricte).

Pour coupler EPROM aux bus du microprocesseur et pour produire les signaux « de service » nous utilisons un PIA. Voici donc une nouvelle application du montage décrit précédemment. Nous avons besoin, au niveau de la plaquette EPROM, des signaux suivants :

\overline{CS}	sélection modes lecture-écriture
PD/PGM	signal de programmation
Horloge H	pour le compteur d'adresses
RES	signal mettant à zéro le compteur

Ces signaux sont émis vers la plaquette EPROM. Une ligne particulière sera utilisée en lecture, par le PIA, pour détecter le basculement de l'interrupteur « haute tension ».

Le synoptique simplifié, le schéma détaillé et un organigramme de fonctionnement aideront à mieux comprendre l'approche du problème.

Avec ce programme, on ne peut traiter que des EPROM vierges au départ. Néanmoins, moyennant quelques modifications simples, il est possible d'accéder à une zone particulière de la mémoire, mais par « pages » de 256 bits.

L'essentiel est de bien comprendre le principe de fonctionnement.

- préparation du PIA
- lecture de la mémoire pour tester sa virginité
- commutation de la « haute tension »
- remise à zéro du compteur d'adresses
- impulsion de programmation
- lecture de la donnée programmée pour comparaison
- incrémentation du compteur d'adresses.

Les affectations des lignes PIA sont :

PORT A : Données (8 bits)
 PORT B: PB0 en sortie CS
 PB1 en sortie PD/PGM
 PB2 en sortie horloge compteur
 PB3 en entrée présence HT
 PB 4 en sortie reset compteur

Le compteur utilisé est un compteur binaire 12 bits, C-MOS (CD 4040), permettant d'adresser jusqu'à 4096 emplacements, ce qui autorise la programmation des 2532.

Le schéma représente le câblage pour une mémoire de type 2516 (2048 octets). La ligne Q12 du compteur n'est pas utilisée dans ce cas. Elle serait reliée à la ligne A11 dans le cas d'une mémoire 4096 octets.

Quelques commentaires sur le programme :

Utilisation mixte du BASIC et du langage machine, dans un but de simplification, car on aurait très bien pu concevoir tout le programme en langage machine.

Selon le type de l'EPROM utilisée, on va modifier le nombre d'adresses à programmer (c'est le rôle des lignes 9210 à 9240 du listing BASIC).

Des emplacements mémoire particuliers ont été utilisés pour indiquer les différents « états » de défauts. En 16535 et 16536 on trouve l'adresse défectueuse, en 16537 son contenu. 16538 indique le type de défaut.

Le sous-programme en langage machine est présenté sur la liste d'assemblage.

CLOCK est la routine qui permet de générer les impulsions d'horloge faisant progresser le compteur d'adresses.

DELA1 exécute un délai de 50 ms utilisé lors de la création de l'impulsion PD/PGM de programmation.

DEFAUT charge les emplacements mémoire utilisés comme « mots d'états » (16535 à 16538) en fonction du contexte.

INIPORB initialise les différentes lignes du PORT B (reportez-vous à la programmation du PIA).

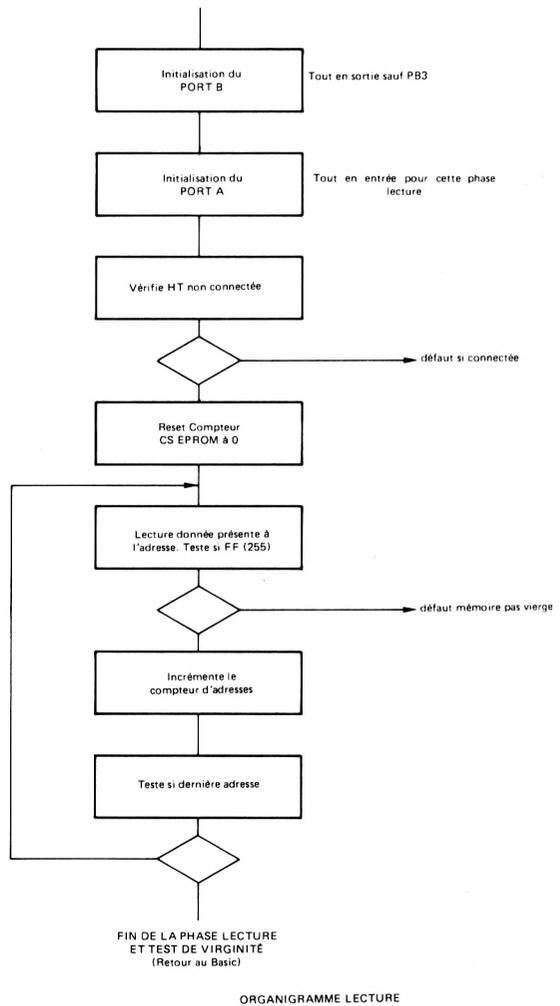
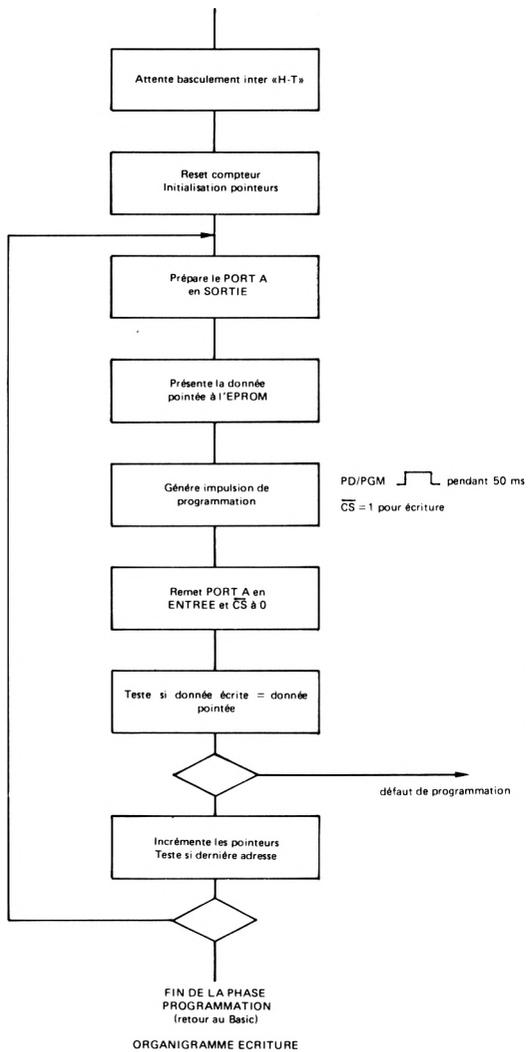
VERIFOX teste la position de l'inter HT qui ne doit pas être sur 25 V pendant la lecture.

RES-CS exécute un reset du compteur d'adresses et met le \overline{CS} à zéro pour la lecture de l'EPROM.

LECTURE lit l'octet et vérifie qu'il est bien égal à 255 (FF), prouvant que la mémoire a été bien effacée.

FIN LECT remet les adresses à zéro, \overline{CS} à 1 et provoque le retour au BASIC.

PASVIERGE Si la mémoire est mal effacée, on met 2 comme code d'erreur, pour renseigner la partie BASIC ...



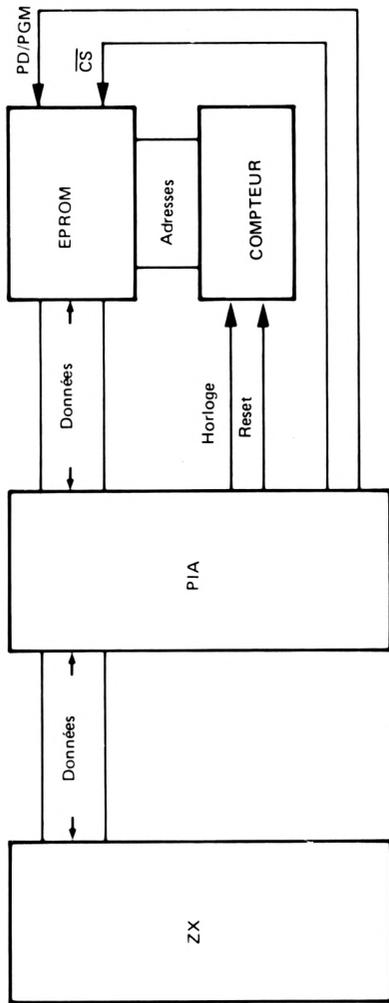


FIGURE 31: SYNOPTIQUE DU PROGRAMMATEUR

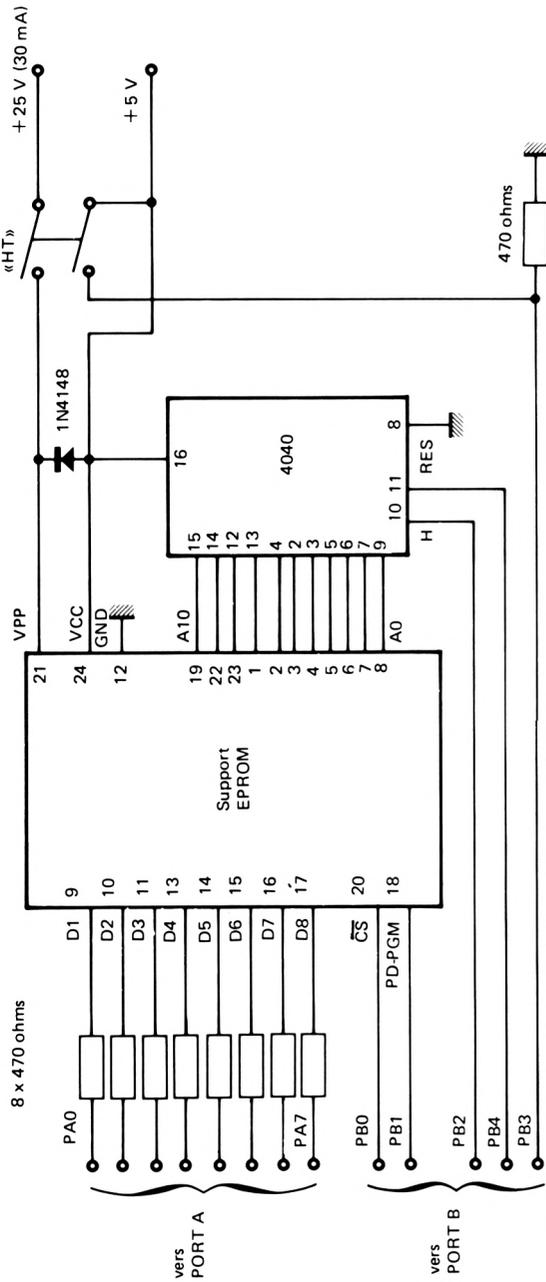


FIGURE 32: SCHEMA DU PROGRAMMATEUR D'EPROM.

Le support d'EPROM est représenté câblé pour une 2516.
Il est souhaitable d'utiliser un support à force d'insertion nulle.

ATTHT on attend que l'opérateur bascule l'inter sur HT pour commencer la programmation. Cette attente s'effectue en testant le bit 3 du PORT B du PIA qui passe à 1 sur HT.

Noter la présence du test de la touche BREAK (CALL \$ F46) et du saut à la routine STOP si BREAK (CALL \$ CDC) permettant d'éviter un bouclage du programme en cas de défaut de l'interrupteur ou d'une liaison. La touche BREAK est également testée lors de la routine CLOCK, parcourue en lecture et en écriture, ce qui permet à tout moment d'interrompre éventuellement le programme.

Ces précautions doivent être prises dans des routines en langage machine car le BREAK ne serait pas possible sans elles.

Dès que l'inter HT a été manipulé, on remet le ZX en mode rapide (CALL \$F23).

INIPOINT initialise les 2 pointeurs. HL lit la zone mémoire où sont stockées les données à programmer et BC l'adresse de l'octet à programmer (au niveau de l'EPROM).

PORASORT prépare le PORT A en sortie pour envoyer l'octet à programmer.

PD/PGM—CS envoie l'impulsion de programmation pendant que CS est à 1.

VALIDITE remet le PORT A en entrée pour relire la donnée programmée et la comparer avec la donnée à programmer. Si elles sont identiques, la programmation de l'octet est correcte.

INCREMENTE incrémente les deux pointeurs et vérifie si on a atteint la dernière adresse.

PASIDENT met 4 comme code d'erreur pour informer le BASIC d'une erreur de programmation.

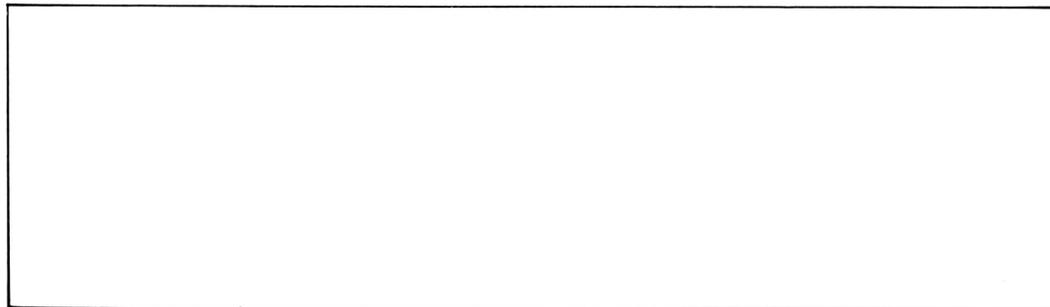
Les séries de 4 NOP servent à faire un petit délai de 5 μ s (stabilisation signaux).

Les données à programmer doivent être rangées dans une zone mémoire protégée (espace 8 K — 16 K ou supérieur à 32 K pour les possesseurs de RAM supplémentaires, ou au-dessus de RAMTOP) avant le chargement du programme. L'adresse de début de cette zone devra être connue et indiquée lors de la question :

« ADRESSE DE LA ZONE MERE ? »

L'alimentation 25 V peut être remplacée (dans le cas d'utilisations peu fréquentes) par 3 piles de 9 V en série et une « stabilisation » par diode zener à 24 V ou 25 V.

Le temps de programmation est d'environ 100 secondes pour 2 K octets.




```

20 REM EPROM-V01
30 GOTO 9200
9100 REM INITIALISATIONS ERREURS
9110 POKE 16535,0
9120 POKE 16536,0
9130 POKE 16537,0
9140 POKE 16538,0
9150 RETURN
9200 PRINT "PROGRAMMATION EPROM"

9210 PRINT "CAPACITE DE L'"EPROM
9220 PRINT " : OCTETS : " - 4 " : " - 1024 " : " - 8 " : " : " - 204
9230 INPUT C
9235 PRINT C
9240 POKE 16557,C
9245 POKE 16795,C
9250 PRINT "ADRESSE DE LA ZONE
MERE ?"
9260 INPUT C
9270 POKE 16707,C-256*INT (C/256)
9275 POKE 16708,INT (C/256)
9280 PRINT "N/L QUAND TOUT EST
PRET"
9285 PAUSE 4E4
9290 POKE 16437,255
9300 REM INITIALISATION LECTURE
9310 FAST
9320 GOSUB 9100
9330 RAND USA 16550
9340 CLS
9350 IF PEEK 16538<>0 THEN GOTO
9360 SLOW
9370 PRINT "INTER HT SUR ON",,,,
ALIM 25V EN SERVICE"
9380 GOSUB 9100
9390 RAND USA 16671
9400 CLS
9410 IF PEEK 16538<>0 THEN GOTO
9420 PRINT "PROGRAMMATION TERMINE"
9430 STOP
9440 STOP
9500 IF PEEK 16538=1 THEN PRINT
"INTER HT MAL POSITIONNE"
9510 IF PEEK 16538=2 OR PEEK 165
=4 THEN PRINT "L'"ADRESSE : ";P
EEK 16535+256*PEEK 16536,"CONTIE
NT : ";PEEK 16537
9520 IF PEEK 16538=2 THEN PRINT
"LA MEMOIRE N'"EST PAS VIERGE"
9530 IF PEEK 16538=4 THEN PRINT
"DEFAULT DE PROGRAMMATION ICI..."
9600 STOP
9998 SAVE "EPROM"
9999 RUN

```

LISTING BASIC DU PROGRAMME «EPROM»

La première ligne contenant le langage machine n'a pas été listée pour éclaircir le listing BASIC.

```

03 REM EPROM...U01
04 REM 16535/16550/16671
05 REM $8
50 REM (
51 REM JP L17
54 REM #XXXXXXXX
55 REM :L5LD A.4
60 REM LD (15002).A
65 REM NOP;NOP;NOP;NOP
70 REM LD A.0
75 REM LD (15002).A
80 REM CALL$F46
85 REM CALLNC.$CDC
90 REM RET
95 REM #XXXXXXXX
100 REM :L10LD DE.6000
105 REM :L15DEC DE
110 REM LD A.D
115 REM OR E
120 REM JR NZ.L15
125 REM RET
130 REM #XXXXXXXX
135 REM :L20LD (16535).8C
140 REM LD (16537).A
150 REM LD A.D
160 REM LD (16538).A
170 REM RET
180 REM #XXXXXXXX
200 REM :L17LD HL.15003
210 REM LD A.0
230 REM LD (HL).A
240 REM LD A.247
250 REM DEC HL
260 REM LD (HL).A
270 REM INC HL
280 REM LD A.4
290 REM LD (HL).A
300 REM LD A.1
310 REM DEC HL
320 REM LD (HL).A
340 REM #XXXXXXXX
350 REM DEC HL
355 REM LD A.0
360 REM LD (HL).A
370 REM DEC HL
380 REM LD (HL).A
390 REM INC HL
400 REM LD A.4
410 REM LD (HL).A
440 REM #XXXXXXXX
450 REM INC HL
460 REM BIT 3.(HL)
470 REM JR Z.L23
480 REM LD D.1
490 REM JP L20
495 REM #XXXXXXXX
500 REM :L23NOP
510 REM LD A.16
520 REM LD (HL).A
530 REM NOP;NOP;NOP;NOP
590 REM #XXXXXXXX

```

```

010 REM LD BC.00
020 REM :L30LD A.(15000)
030 REM CP 255
040 REM JR NZ.L32
050 REM CALLL5
060 REM INC BC
070 REM LD A.B
080 REM CP 8
090 REM JR NZ.L30
100 REM *PUSHB
110 REM LD A.17
120 REM LD (15002).A
130 REM RET
140 REM *PUSHB
150 REM :L32LD D.2
160 REM JP L20
170 REM *PUSHB
180 REM LD HL.15002
190 REM LD A.17
200 REM LD (HL).A
210 REM NOP;NOP;NOP;NOP
220 REM :L45CALL$F46
230 REM CALLNC.#CDC
240 REM BIT 3.(HL)
250 REM JR Z.L45
260 REM CALLL10
270 REM BIT 3.(HL)
280 REM JR Z.L45
290 REM LD A.19
300 REM LD (16507).A
310 REM CALL$F23
320 REM *---PROGRAMMATION---
330 REM *PUSHB
340 REM LD HL.00
350 REM LD BC.00
360 REM *PUSHB
370 REM :L50LD A.0
380 REM LD (15001).A
390 REM LD A.255
400 REM LD (15000).A
410 REM LD A.4
420 REM LD (15001).A
430 REM LD A.(HL)
440 REM LD (15000).A
450 REM NOP;NOP;NOP;NOP
460 REM *PUSHB
470 REM LD A.(16507)
480 REM LD (15002).A
490 REM CALLL10
500 REM LD A.(16507)
510 REM AND $11
520 REM LD (15002).A
530 REM NOP;NOP;NOP;NOP
540 REM *PUSHB
550 REM LD A.(16507)
560 REM AND $10
570 REM LD (15002).A
580 REM LD A.0
590 REM LD (15001).A

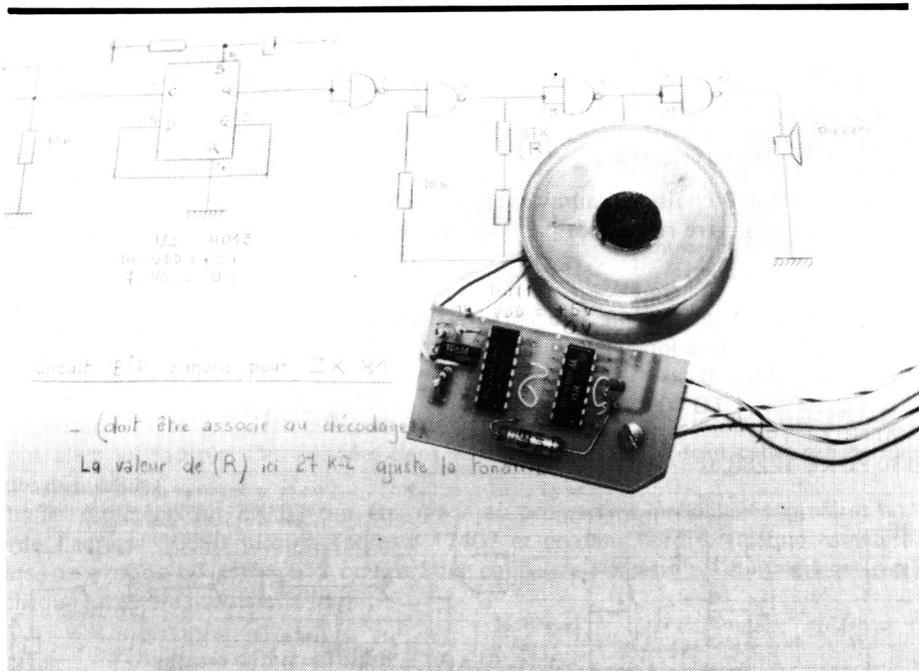
```

```

13000 REM LD (15000).A
13010 REM LD A.4
13020 REM LD (15001).A
13030 REM LD A.(15000)
13040 REM CP (HL)
13050 REM JR NZ.L60
13060 REM *FIN DE PROGRAMME
13070 REM CALL L5
13080 REM INC HL
13090 REM INC BC
13095 REM LD A.3
13098 REM LD (16507).A
14000 REM LD A.B
14100 REM CP 8
14200 REM LR NZ.L50
14300 REM LD A.17
14400 REM LD (15002).A
14500 REM RET
15000 REM *FIN DE PROGRAMME
15100 REM :L60 LD D.4
15200 REM JP L20
19999 REM )

```

Cette liste d'assemblage du programme «EPROM» est en fait celle qu'on obtient en utilisant l'assembleur ZX AS commercialisé en France chez l'importateur. Cet assembleur, malgré quelques tares, est en réalité d'un grand secours pour qui désire écrire un long programme en langage machine.



LE BIP SONORE (voir page suivante).

MONTAGE D'UN CIRCUIT BIP SONORE

Il peut être utile, dans certains cas, de disposer d'un signal sonore avertissant l'opérateur que les opérations en cours sur la machine ont atteint un certain stade. D'autres applications sont possibles, notamment dans les jeux ou encore ... pour faire de la télégraphie MORSE.

Ce petit montage extrêmement simple fait appel à deux circuits intégrés courants (C-MOS).

une bascule CD 4013
une NAND CD 4011

L'oscillateur est constitué, de façon classique, par deux portes NAND. La fréquence d'oscillation BF peut être changée en modifiant la valeur de la résistance (R) de 27 kΩ.

La sortie de l'oscillateur attaque un petit transistor transducteur sonore (résonateur piezo, buzzer, petit haut-parleur de 50 Ω).

L'oscillateur est commandé par une bascule qui sert à le mettre en fonction et à l'arrêter.

L'entrée de la bascule est reliée, pour un OU à diodes, au signal d'écriture (WR) et à une sortie du décodeur d'adresses (ici la sortie Y2 du décodeur 74LS138 décrit Figure 9).

La mise en fonctionnement et l'arrêt du BIP sont réalisés simplement par une opération d'écriture en mémoire.

Avec le câblage sur la sortie Y2 du 74LS138, le circuit répond à toute adresse située dans la plage s'étendant de 12288 à 14335. L'utilisation du signal WR fait que sa mise en fonctionnement ne s'effectuera que sur un ordre POKE.

Pour vérifier facilement le fonctionnement du circuit, il suffira de faire

POKE 12288, 0

ce qui a pour effet de le mettre en marche

POKE 12288, 0

pour l'arrêter.

Noter que la valeur 0 choisie ici aurait pu être quelconque, dans la plage 0 à 25, ainsi que la valeur 12288 qui peut être remplacée par toute adresse située entre 12288 et 14335, par exemple 13000.

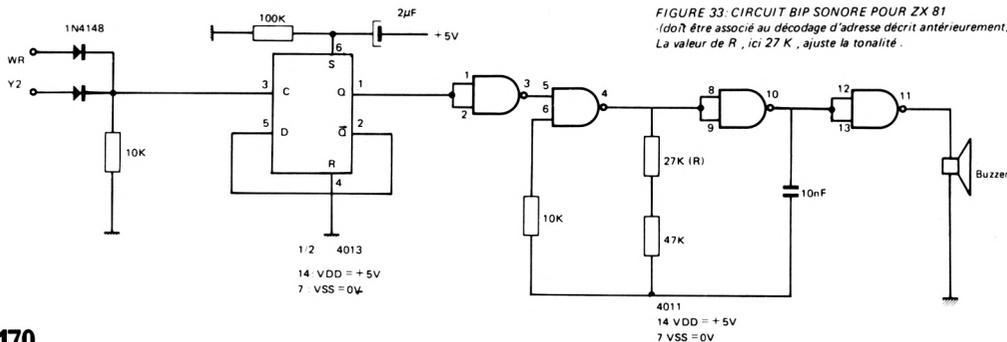
L'utilisation dans un programme fera appel à deux instructions POKE séparées par un petit délai, réglé en fonction de la longueur du signal.

100 POKE 12288, 0

110 FOR T = 1 TO 10

120 NEXT T

130 POKE 12288, 0



.MODIFICATION ET AMELIORATION DU GRAPHISME.

Pour les possesseurs de l'imprimante SINCLAIR, voici un petit programme qui leur permettra de créer eux-mêmes leurs propres caractères. Un exemple est la possibilité d'imprimer des minuscules. En fait, il est tout-à-fait possible de recréer les 64 caractères du ZX 81.

Le principe de ce programme réside dans une petite modification de la routine COPY de la ROM BASIC. Cette routine se situe dans la ROM entre l'adresse 2153 et 2292. En utilisant le désassembleur, il est facile de s'apercevoir qu'à l'adresse 2116 il est fait appel au générateur de caractères en chargeant dans H la valeur 15H qui, après une rotation à gauche des bits du registre, donnera 30H (adresse haute du générateur de caractères). En modifiant cette valeur, il est alors possible d'utiliser tout autre générateur qui pourra alors se situer en RAM.

Bien sûr, la routine COPY se trouvant dans la ROM, il n'est pas possible d'y faire des modifications. La solution va être de la transférer en RAM.

Le premier travail est de créer un REM de 1000 octets, grâce au programme décrit précédemment, et de taper le programme suivant :

```

2 REM
5 FAST
10 LET L=16514
20 FOR N=2153 TO 2292
30 POKE L,PEEK N
40 LET L=L+1
50 NEXT N
60 POKE 16576,33
70 LET L=16896
80 FOR N=7680 TO 8191
90 POKE L,PEEK N
100 LET L=L+1
110 NEXT N
120 SLOW
130 REM CODE DES MINUSCULES
140 FOR N=17200 TO 17407
150 INPUT A
160 SCROLL
170 PRINT N; "-"; A
180 POKE N,A
190 NEXT N
    
```

Transfert routine ROM en RAM à partir de 16514

Modification adresse générateur caractères

Transfert générateur caractères ROM en RAM

Entrée codes caractères minuscules

Faire ensuite RUN et, après quelques instants en mode FAST, le ZX attendra des données que vous allez lui fournir. Ces données dans l'exemple ci-dessous sont celles qui permettent de créer des minuscules.

Tous les caractères du ZX 81 ont été, grâce au programme précédent, transférés en RAM à partir de l'adresse 16896 jusqu'à l'adresse 17407 et ce dans l'ordre. Chaque caractère occupe 8 octets, on a donc $64 \times 8 = 512$ octets. Pour connaître l'adresse où commencent ces 8 octets pour chaque caractère, il suffit de faire :

$$\text{adresse début} = 16896 + (X * 8)$$

code Sinclair du caractère

Dans notre exemple, comme il s'agit de créer des minuscules à partir de la lettre A, on va donc avoir :

$$\text{adresse début} = 16896 + (38 \times 8) = 17200$$

$$\text{adresse fin} = 17200 + (26 \times 8) = 17408$$


 26 lettres

Nous vous décrivons plus loin un programme basic qui vous permettra aisément de définir vos caractères en les visualisant agrandis sur votre écran.

Voici donc les données permettant d'obtenir des minuscules avec l'imprimante :

codes des minuscules de a à z

172000 - 0	0	14	1	15
172005 - 17	15	0	16	16
172010 - 30	17	17	17	30
172015 - 0	0	0	15	16
172020 - 16	16	15	0	1
172025 - 1	15	17	17	17
172030 - 15	0	0	0	14
172035 - 17	31	16	14	0
172040 - 2	4	4	14	4
172045 - 4	4	0	0	15
172050 - 17	17	15	1	6
172055 - 0	16	16	30	17
172060 - 17	17	17	0	4
172065 - 0	12	4	4	4
172070 - 14	0	4	0	4
172075 - 4	4	20	0	0
172080 - 0	0	9	10	12
172085 - 10	9	0	12	4
172090 - 4	4	4	4	14
172095 - 0	0	0	26	21
173000 - 21	21	21	0	0
173005 - 0	30	17	17	17
173010 - 17	0	0	0	14
173015 - 17	17	17	14	0
173020 - 0	30	17	17	30
173025 - 16	16	0	0	15
173030 - 17	17	15	1	1
173035 - 0	0	0	11	12
173040 - 0	0	0	0	0
173045 - 0	15	16	14	1
173050 - 30	0	0	4	14
173055 - 4	4	4	2	0
173060 - 0	0	17	17	17
173065 - 17	15	0	0	0
173070 - 17	17	10	10	4
173075 - 0	0	0	17	17
173080 - 17	21	10	0	0
173085 - 0	17	10	4	10
173090 - 17	0	0	17	17
173095 - 17	15	1	0	0
174000 - 0	0	31	2	4
174005 - 0	31	0	128	128

UN GENERATEUR DE CARACTERES PROGRAMMABLES

Nous avons souligné, à maintes reprises, que l'espace inoccupé allant de 8192 à 16384 pouvait fort bien être utilisé. Voici donc le moyen d'ajouter, à peu de frais, un générateur de caractères programmables sur le ZX. A peu de frais, car nous allons utiliser la RAM 1 K interne qui devient inutile lorsque l'on connecte une extension mémoire. Le principe reste le même si vous ne voulez pas utiliser la mémoire 1 K interne, mais vous devrez alors ajouter un boîtier mémoire (4118).

Comme toujours, il faudra réaliser un décodage d'adresses. Reportez-vous aux montages proposés antérieurement dans cet ouvrage (Figure 9, le 74LS138 associé au 74LS32 de la Figure 8). L'avantage de ce système est que le décodage d'adresses est assez complet. On peut évidemment le simplifier ...

La sortie Y_0 (patte 15 du 74LS138) sera reliée au RAMCS (lire les explications et voir Figure 10 pour la modification d'une piste du circuit imprimé pour accéder au RAMCS de la mémoire vive interne).

La RAM sera alors sélectionnée dans la plage 8192 à 10239. Comme elle ne fait qu'un K octet, on n'utilisera que l'espace 8192 à 9215 dans notre générateur de caractères.

Sachant qu'un caractère est défini en 8 octets, nous aurons donc $1024/8 = 128$ caractères programmables : c'est plus qu'il n'en faut !

Examinons la modification à faire subir au ZX pour réutiliser la RAM 1 K. Rappelons que vous devez avoir auparavant modifié (Figure 10) la piste «RAMCS».

Nous considérerons un ZX équipé d'une mémoire 4118. Les possesseurs de ZX équipés de 2 2114 extrapôleront. Il est évident que, si les mémoires sont montées sur supports, la modification n'en sera que plus simple. En effet, il faut pouvoir dégager du circuit imprimé les pattes correspondant aux lignes d'adresses A_0 à A_8 de la mémoire (4118).

Ceci peut se faire aisément en sortant le circuit de son support et en pliant, avec beaucoup de précautions, les pattes 1, 3, 4, 5, 6, 7, 8, 22, 23 du circuit intégré, à l'horizontale.

Souder délicatement sur ces pattes des fils fins (du fil à wrapper fait très bien l'affaire) de longueur d'environ 12 cm. Une fois ce travail exécuté, replacer avec précaution la mémoire sur son support.

Repérer ensuite les œillets du circuit imprimé qui se situent à gauche de la ROM. Aidez-vous pour cela de la Figure 34. Ils correspondent aux lignes d'adressage ($A_0 - A_8$).

Relier les fils venant de la RAM aux œillets en suivant les indications du tableau Figure 35.

Ce montage est rendu nécessaire pour s'affranchir des résistances de $1\text{ k}\Omega$, R18 à R26 et connecter directement notre RAM sur le bus adresses issu du circuit spécialisé.

Le montage équivalent est celui du synoptique Figure 36.

L'accès au générateur de caractères de la ROM ou de la RAM s'effectue sous le contrôle du circuit spécialisé. La sélection de l'un ou de l'autre générateur s'effectue grâce au décodeur d'adresses.

Pour comprendre le fonctionnement, il faut se rappeler que l'adresse basse du générateur de caractères est fournie par le registre I.

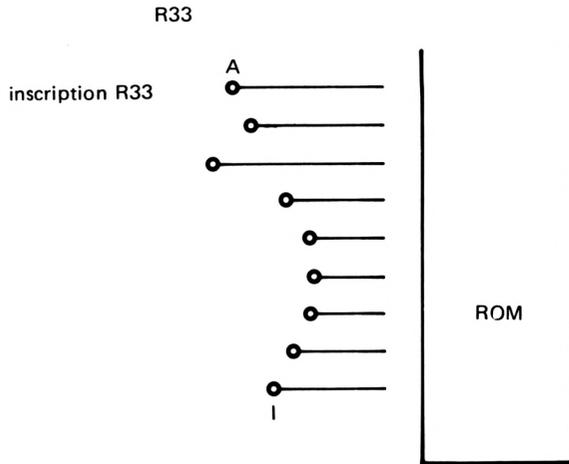


Fig. 34 -- Repérage des œillets du circuit imprimé.

On admettra que l'œillet du haut est l'œillet A. Celui du bas, l'œillet I.

œillet	Patte de 4118	ligne adresse
A	4	A7
B	5	A6
C	3	A8
D	23	A5
E	6	A4
F	22	A3
G	7	A2
H	1	A1
I	8	A0

Fig. 35 -- Tableau des connexions.

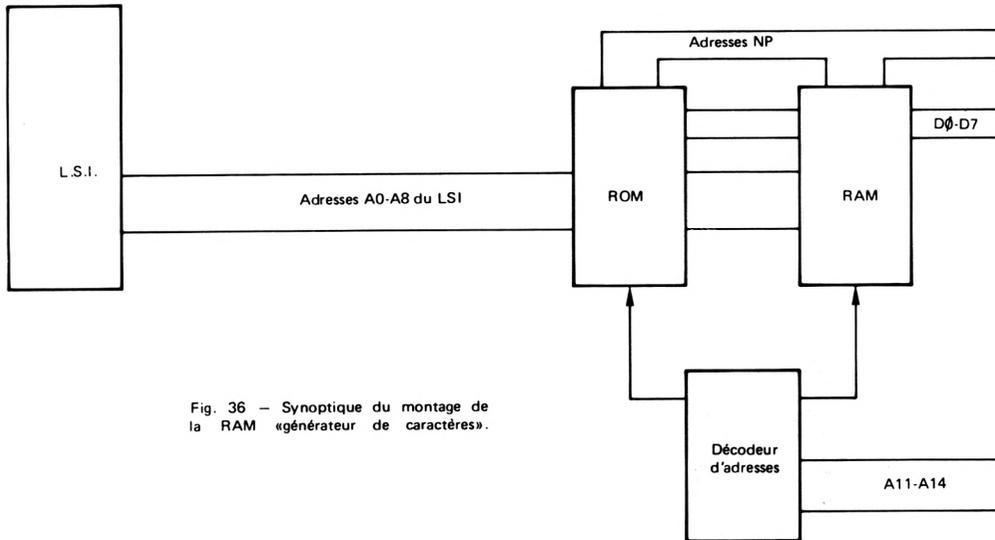


Fig. 36 -- Synoptique du montage de la RAM «générateur de caractères».

En ROM, nous savons déjà que ce générateur commence en 7680, soit en hexadécimal : 1E00. Le registre I contient la valeur 1E (30 en décimal). En modifiant cette valeur, chargée dans le registre I, on peut donc accéder à un autre générateur. Celui qui nous intéresse ici est notre RAM dont l'adresse de début est 8192 = 2000 (4). En mettant 20 (H) = 32 décimal dans le registre I, on y aura accès.

Le jeu de caractères ainsi défini sera bien sûr à charger en mémoire après chaque coupure d'alimentation du ZX. Si on veut éviter cette contrainte, on peut, bien entendu, l'implanter en EPROM ce qui se justifierait dans le cas de caractères fréquemment utilisés (par exemple des symboles électroniques).

PROGRAMMATION – DEFINITION DES CARACTERES

Définition d'un caractère

Chaque caractère est codé sur 8 octets. Pour les caractères non graphiques l'octet 0 et l'octet 7 sont à zéro ; ceci permet de ménager l'interligne horizontal séparant deux lignes d'écran. Vous avez remarqué que les caractères graphiques, eux, se touchent.

Selon les besoins, il sera possible de transférer ou non tout ou partie du générateur ROM en RAM. Par exemple, si on désire obtenir dans un même programme de nouveaux symboles graphiques en même temps que l'alphabet, on transférera les lettres et on remplacera les anciens symboles graphiques, la ponctuation et les chiffres par les nouveaux caractères.

Ainsi, pour des schémas électroniques, on peut conserver les chiffres, créer des diodes, résistances, transistors, condensateurs, orientés différemment, conserver quelques lettres, telles que C, R, T et créer des μ , β , Ω ou des minuscules.

Comment créer un caractère

On devra d'abord le dessiner sur un papier matrice 8 x 8. Chaque point noirci sera égal à UN, les autres points à ZERO. Voici, représenté ci-dessous, le symbole d'une diode et son profil équivalent binaire.

	1	2	3	4	5	6	7	8		1	2	3	4	5	6	7	8	Hexa	Décimal
1		■				■			1	0	1	0	0	0	1	0	0	4 4	68
2		■	■						2	0	1	1	0	0	1	0	0	6 4	100
3		■	■	■					3	0	1	1	1	0	1	0	0	7 4	116
4	■	■	■	■	■	■	■	■	4	1	1	1	1	1	1	1	1	F F	255
5	■	■	■	■	■	■	■	■	5	1	1	1	1	1	1	1	1	F F	255
6	■	■	■						6	0	1	1	1	0	1	0	0	7 4	116
7	■	■	■	■					7	0	1	1	0	0	1	0	0	6 4	100
8									8	0	1	0	0	0	1	0	0	4 4	68

Les 8 valeurs décimales (ou hexadécimales ...) obtenues seront écrites dans le générateur de caractères.

A titre d'exemple, nous allons remplacer le caractère graphique SINCLAIR ■ dont le code est 1 par notre diode.

Auparavant quelques remarques :

- il faut conserver l'espace (code 0) donc les 8 premiers octets de notre nouveau générateur seront à zéro.
- par commodité, dans cet exemple restreint, nous transférerons la ROM en RAM pour conserver les autres caractères.
- comme nous ne créons qu'un seul caractère dans cet exemple, nous le ferons par des POKE successifs. Il est évident que, lors de la création de jeux de caractères complets, il faudra écrire un programme plus élaboré...
- pour modifier le registre I, on utilise une routine en langage machine. Pour revenir au générateur de caractères d'origine, il faudra ré-exécuter cette routine pour remettre la valeur 30 dans I ; ceci est surtout important si vous avez modifié les chiffres et les lettres, car sans cette précaution les listings deviendraient illisibles.
- notre générateur de caractères (1024 octets) pourra contenir 2 fois 64 caractères. Le premier sera adressé quand le registre I contiendra 32 (8192 à 8703). Pour le second (8704 à 9215), le registre I devra être chargé avec la valeur 34.
- rappelons que le transfert de la ROM en RAM n'est pas absolument nécessaire, mais qu'il évite d'avoir à jongler avec les modifications du registre I quand un programme utilise des symboles et des textes simultanément sur l'écran...

```
1 REM .....
5 FAST
10 REM NETTOYAGE RAM
15 FOR N=8192 TO 9215
20 POKE N,0
25 NEXT N
30 REM TRANSFERE LA ROM EN RAM
35 FOR N=7680 TO 8191
40 POKE (N+512),PEEK N
45 NEXT N
```

Ecrivez ce programme et faites RUN.

N'oubliez pas la ligne 1

Repassez en mode lent : SLOW

Faites alors :

POKE 16514,62

POKE 16515,32

POKE 16516,237

POKE 16517,71

POKE 16518,201

vous venez d'écrire la routine de modification du registre I dont voici la traduction en assembleur:

```
LD A, 32
LDI, A
RET
```

Pour définir le profil de la diode, faites alors :

```
POKE 8200, 68
POKE 8201, 100
POKE 8202, 116
POKE 8203, 255
POKE 8204, 244
POKE 8205, 116
POKE 8206, 100
POKE 8207, 68
puis RAND USR 16514
```

PRINT CHR\$ 1 fera apparaître le symbole de la diode. Pour retrouver le caractère graphique d'origine faites :

```
POKE 16515, 30
RAND USR 16514
PRINT CHR $ 1
```

Le programme BASIC de transfert de la ROM en RAM ou la partie «nettoyage» de la RAM pourraient être écrits en langage machine. L'exécution est alors instantanée.

Transfert de la ROM dans la première page RAM

```
LD BC, 512
LD HL, 7680
LD DE, 8192
LDIR
RET
```

Nous utilisons l'instruction LDIR qui effectue le transfert d'un bloc d'octets de l'adresse désignée par HL (ici 7680) à celle pointée par DE (ici 8192). Le nombre d'octets à transférer est indiqué par BC (ici 512).

Nettoyage de la RAM (on écrit des zéros de 8192 à 9215).

```
LD BC, 1024
LD HL, 9215
«L1» DEC HL
LD (HL), 0
DEC BC
LD A, B
CP 0
JRNZ «L1»
LD A, C
CP 0
JRNZ «L1»
RET
```

Notons que, pour tester le passage à zéro d'un registre double dont on décrémente le contenu, il y a une autre programmation possible, plus courte :

```
LD BC, 1024
«L1» DEC BC
LD A, B
OR C
JR NZ «L1»
RET
```

Ces exemples doivent vous permettre de voir qu'il est souvent aisé de passer du BASIC au langage machine.

A titre d'exemple, vous trouverez un schéma électronique dessiné au moyen des caractères programmables.

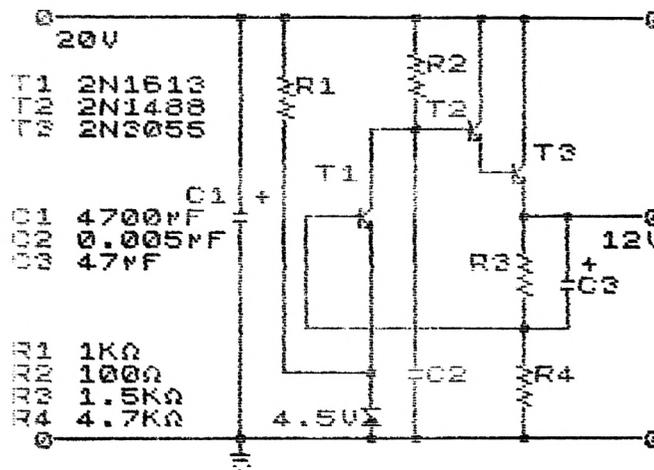
En face de chaque symbole dans la liste, vous trouverez le code du caractère correspondant.

Exemple : PRINT CHR \$ 2 donne la lettre grecque mû (μ).

Les codes autorisés vont de 0 à 63.

Les caractères en vidéo inversée s'obtiennent en ajoutant 128 au code du caractère. PRINT CHR \$ 130 affichera la lettre mû en vidéo inversée.

Groupés 8 par 8, vous trouverez également la liste des octets correspondants à ces caractères, écrits par POKE entre 8192 et 8711.



3		1	+
5	7	3	7
4	8	5	8
6	9	7	9
8	+	9	+
10	1	11	H
12	K	13	K
14	+	15	+
16	+	17	+
18	+	19	+
20	1	21	+
22	-	23	-
24	3	25	r
26	2	27	.
28	0	29	1
30	2	31	3
32	4	33	5
34	6	35	7
36	8	37	9
38	+	39	+
40	0	41	D
42	E	43	F
44	L	45	+
46	D	47	D
48	K	49	L
50	2	51	N
52	+	53	+
54	7	55	R
56	L	57	T
58	+	59	U
60	-	61	+
62	L	63	+

0	0	0	0	0100	0	0	0
0	0	0	0	0000	0	0	0
0	0	06	52	0040	02	02	0
04	96	94	2003	0216	96	64	0
50	60	74	1450	0217	17	10	4
236	10	41	6900	0232	69	40	16
127	02	16	0000	0240	62	127	0
0	15	0	2550	0256	15	0	0
04	00	46	2550	0256	00	04	0
0	02	64	2550	0254	02	0	0
24	40	72	1000	0270	0	0	0
200	240	192	1900	0210	240	200	0
136	144	160	1900	0210	160	152	164
136	144	160	1900	0210	164	176	160
0	0	20	2550	0200	0	0	0
0	0	20	2550	0310	0	0	0
0	0	20	0300	0200	0	0	0
0	4	2	2550	0200	4	0	0
0	0	20	2550	0200	0	0	0
0	0	20	2550	0344	0	0	0
0	0	0	0000	0300	0	0	0
0	0	0	0000	0300	0	0	0
0	0	0	0000	1000	0	0	0
00	126	0	1950	0370	0	0	0
00	60	170	1700	0304	17	0	0
0	0	0	1400	0390	0	0	0
0	40	12	0400	0100	40	12	0
0	0	0	0000	0400	24	24	0
0	60	70	7400	0416	90	60	0
0	24	40	0400	0404	0	02	0
0	60	66	0400	0200	02	126	0
0	60	66	1200	0440	02	60	0
0	0	24	4000	0472	126	0	0

0	126	64	124	66	60	0
0	60	64	124	66	60	0
0	126	2	4	16	32	0
0	60	66	60	66	60	0
0	60	66	66	2	60	0
127	0	20	62	0	0	0
0	0	0	127	28	0	127
0	60	66	64	66	60	0
0	120	68	66	68	120	0
0	126	64	120	64	126	0
0	126	64	120	64	64	0
1	2	4	248	0	0	0
0	0	112	127	0	0	0
252	130	129	129	130	252	0
252	66	33	33	66	252	0
0	68	72	112	68	66	0
0	64	64	64	64	126	0
0	24	36	66	36	102	0
0	66	98	82	70	66	0
0	36	36	231	36	0	0
0	0	62	0	62	0	0
0	0	0	248	0	0	0
0	126	66	66	68	66	0
0	0	0	248	0	0	0
0	254	16	16	16	16	0
0	100	116	255	100	68	0
0	34	34	34	20	0	0
0	0	0	255	0	0	0
0	0	112	127	32	64	126
0	0	0	15	0	0	0
0	127	0	62	33	0	0
0	150	0	255	255	0	255

CONCLUSIONS

Avant de réaliser un programme, il faut évidemment avoir une bonne idée de ce que l'on veut faire et des possibilités de la machine.

Un programme se conçoit en plusieurs étapes. Dans la première, on trace un plan général du programme: La seconde sera une analyse beaucoup plus fine du séquençement, qui conduira à l'élaboration d'un organigramme détaillé. Ceci est très important au moment de la mise au point du programme, lorsque les résultats sont différents de ceux qui étaient attendus. Il est plus facile de suivre un organigramme sur papier qu'un listing. Cette remarque trouve sa pleine justification lors de l'élaboration d'un programme en langage machine.

Autre avantage de l'organigramme, il permet de reconduire un programme développé sur une machine avec un type de langage donné sur une autre machine ne possédant pas forcément le même jeu d'instructions.

La phase finale passe par l'écriture du programme avec le clavier. On respectera un espacement suffisant entre pas de test, ce qui permettra par la suite, lors de la mise au point, d'insérer éventuellement des lignes de modification.

Quelques commentaires, judicieusement placés dans le listing, doivent permettre de reprendre en main le programme sans avoir à rechercher «ce qu'on a voulu faire», quelques mois plus tard, sans compter que ces commentaires guideront un éventuel lecteur.

A partir de ces quelques informations et des programmes de démonstration proposés, il ne vous reste plus qu'à développer vos propres logiciels, en essayant de tirer au mieux parti des possibilités «cachées» du ZX (routines de la ROM, système, etc...).

Les applications sont multiples et, à titre expérimental, vous pourrez même essayer (n'abusez pas, car les transmissions de données sur les bandes amateurs ne sont pas encore autorisées) de transmettre et recevoir des programmes. Cela est tout-à-fait possible, à condition qu'il n'y ait pas le moindre brouillage. De plus, il faut effectuer un couplage direct entre le micro-ordinateur (ou le magnétophone) et l'émetteur-récepteur.

Dans un souci de simplification et pour éviter aux lecteurs un travail de frappe à la fois inutile et fastidieux, tous les programmes décrits dans cet ouvrage sont disponibles sur deux cassettes.

La première contient CW et RTTY

La seconde les autres programmes

QUELQUES TRADUCTIONS D'ABREVIATIONS UTILISEES

ROM	Read Only Memory (mémoire «morte»)
EPROM	Erasable Programmable Read Only Memory (Mémoire à lecture seule, programmable et effaçable)
RAM	Random Access Memory (mémoire à lecture - écriture)
PIA	Periphal Interface Adapter (circuit d'interface pour adaptation des périphériques)
ACIA	Asynchronous Communications Interface Adapter (Circuit pour liaisons séries adaptation communications asynchrones)
UART	Universal Asynchronous Receiver Transmitter (circuit émetteur-récepteur pour liaisons asynchrones série/parallèle)

SOMMAIRE

PREMIERE PARTIE

Introduction	11
Architecture du système	13
Limites du ZX	16
Périphériques	19
Les extensions	21
Le Basic du ZX	27
Les variables systèmes	31
Introduction au langage machine	34
Le microprocesseur Z-80	40
Le jeu d'instructions	44

DEUXIEME PARTIE

Les programmes «utilitaires»	55
– désassembleur	57
– création automatique de REM	65
– transfert d'un EPROM en RAM	68
Les programmes à vocation radio	70
– programme de transmission d'images	70
– émission-réception code morse	80
– calcul QTH Locator et Azimut	102
– suivi de contest	106
– calculs divers	111
– programme «MIRE»	117
– programme QSO TV	119
– programme Fidep	121
– programme émission-réception RTTY	123
– programme «Fichier 2»	146

TROISIEME PARTIE

Amélioration des entrées-sorties	151
– PIA	151
– le port d'entrée K7 du ZX 81	156
– la carte 8 entrées - 8 sorties	157
– programmeur d'EPROM	160
– montage d'un circuit Bip sonore	170
Modification et amélioration du graphisme	171
– graphisme sur imprimante	171
– un générateur de caractères programmables	174

CONCLUSIONS

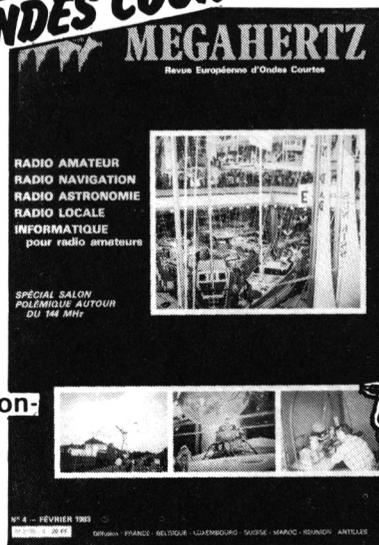
183

CHAQUE MOIS, N'OUBLIEZ PAS:

MEGAHERTZ

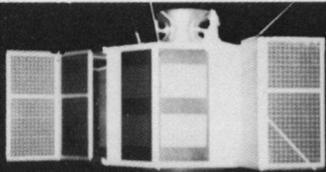


LA REVUE EUROPEENNE D'ONDES COURTES



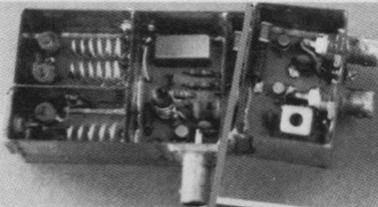
***Pour ne pas l'oublier, ABONNEZ VOUS !**

**MEGAHERTZ: SORACOM- 16, av. Gros Malhon-
35000 RENNES- Tél: (99) 54. 22. 30.**



**FLORENCE
MELLET**

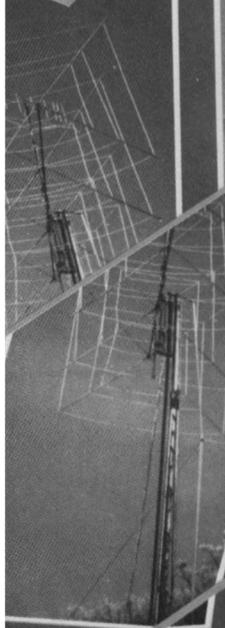
**SYLVIO
FAUREZ**



TECHNIQUE RADIO TECHNicien L'AMATEUR POUR SUIVRE A NOUS!..

p

GR



**EDITIONS
SORACOM**

**H&I
RABT**

ALLIANCE
EX "FRED A. "FRED" LAUN
4220
1928 HIKAL ISLAND (USA)



Composition - Maquette : SORACOM
Impression JOUVE - Mayenne
N° d'éditeur : 015
N° 12258. Dépôt légal : Janvier 1984



Prix de vente : 90 F

SORACOM - informatique



IMPRIME EN FRANCE

COMMUNIQUEZ VOTRE ZAX 81 DENIS BOYD - EDITEUR

AMSTRAD

CPC



MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA



<https://acpc.me/>

[FRA] Ce document a été préservé numériquement à des fins éducatives et d'études, et non commerciales.

[ENG] This document has been digitally preserved for educational and study purposes, not for commercial purposes.

[ESP] Este documento se ha conservado digitalmente con fines educativos y de estudio, no con fines comerciales.