Denis BONOMO

Eddy DUTERTRE

COMMUNIQUEZ AVEC VOTRE ORIC 1 ET VOTRE ATMOS

SORACOM éditions
INFORMATIQUE

COMMUNIQUEZ AVEC VOTRE ORIC 1 ET VOTRE ATMOS

Denis BONOMO

Eddy DUTERTRE

COMMUNIQUEZ AVEC VOTRE ORIC 1 ET VOTRE ATMOS

SORACOM éditions
INFORMATIQUE

INTRODUCTION

Des millions de micro-ordinateurs en service dans le monde. Le votre est là, devant vous, car vous avez décidé de participer, à votre niveau à cette « révolution informatique ».

Comme pour toute autre chose, l'abus de l'informatique peut conduire à des excès et des résultats regrettables. Délaissons ce côté négatif pour entreprendre l'examen des différentes possibilités qui s'offrent à vous au moyen du clavier magique.

La communication, de nos jours, passe de plus en plus par l'informatique et ce, à tous les niveaux. Vous pourrez très prochainement accéder depuis votre fauteuil à une foule d'informations simplement en composant un numéro de téléphone. Souhaitons que, d'ici quelques mois, les utilisateurs de radio que nous sommes, soient autorisés à en faire de même et à procéder à des échanges de données, à but scientifique, sur nos fréquences.

Chacun pourra mettre au service des autres ses compétences propres dans un domaine particulier. Certains proposeront les éphémérides complétées de prévisions de passages de satellites, d'autres nous fourniront des informations sur la propagation, le trafic DX ou tout autre événement touchant le cercle des amateurs.

Pour être à même de participer à ces activités, il sera bon, voire indispensable, de posséder quelques connaissances en micro-informatique pour concevoir, modifier ou simplement utiliser les logiciels appropriés.

Le micro-ordinateur individuel n'a cessé d'évoluer et gageons que ses capacités progresseront encore rapidement dans les prochains mois.

Dans l'instant préoccupons-nous de la machine que nous avons choisie : l'ORIC-1.

AVERTISSEMENT

Les programmes décrits dans cet ouvrage ont été conçus pour l'ORIC-1 avec BASIC V1-∅. Le lecteur désireux de les adapter sur BASIC V1-1 (ATMOS) devra se reporter au dernier chapitre du livre où sont données les équivalences entre les adresses des deux ROM et les variables système.

D'une manière générale, les programmes devant subir des modifications importantes ont été listés à nouveau. Nous laissons au lecteur le soin d'adapter les petits exemples sur ATMOS.

Les auteurs.

PRÉSENTATION DE L'ORIC-1

- SYSTÈME MINIMUM REQUIS
- STRUCTURE DE L'ORIC-1
- POSSIBILITÉS D'EXTENSION
- UTILISATION DE LA MÉMOIRE

PRÉSENTATION DE LA MACHINE

Nous souhaitons rendre cet ouvrage attrayant, aussi bien pour le programmeur expérimenté que pour le débutant. C'est à l'intention de ce dernier que nous décrirons en détail l'ORIC et le système minimum qui doit l'entourer.

Comme tout travail de programmation doit passer, auparavant, par une phase d'analyse, il sous-entend également une bonne maîtrise des possibilités (nous dirons aussi des « ressources ») matérielles de la machine.

C'est le lien qui existe entre le matériel (hardware ou « hard » pour les anglo-saxons et... le milieu informatique sympatisant) et le logiciel (software ou « soft »). Notons au passage l'impression un peu péjorative que laisse l'emploi des termes « soft » et « hard » reléguant le matériel à l'état de simple quincaillerie.

SYSTÈME MINIMUM

Le minimum nécessaire se composera donc de :

— la machine (version 48 K à préférer à la version 16 K vu la différence de prix et les difficultés de modification ultérieure pour passer de 16 à 48 K),

— le magnétophone à cassettes,

— le téléviseur.

Nous allons nous attarder un petit peu sur ces deux derniers composants de la chaîne car ils sont importants.

Le magnétophone : tout système à cassettes n'est pas d'une fiabilité absolue à cause, justement, de la qualité du magnétophone et des cassettes utilisées. Le fonctionnement sera pratiquement toujours correct quand on utilise des cassettes enregistrées et lues par le même magnétophone. Les choses se gâtent quand on passe les cassettes d'un magnétophone à un autre.

Les problèmes sont dus aux différences de vitesse de défilement et d'azimutage des têtes magnétiques. Il est relativement aisé de résoudre le premier en fonctionnant toujours sur alimentation secteur, le circuit de régulation, de vitesse faisant le reste. Pour l'azimutage, c'est plus délicat, c'est pourquoi nous préconisons l'utilisation d'un magnétophone où le réglage d'azimutage est accessible. Le plus souvent, il se présente sous la forme d'une petite vis de réglage apparaissant au niveau de la tête de lecture. Il sera toujours possible de retoucher au réglage en cas de besoin.

L'effet de ce réglage se fait sentir à l'audition. On considéra que le réglage est bon lorsque le son sera au maximum aigu, ce qui correspondra en général, à un volume maximum. Ce réglage agissant sur les positions respectives des axes tête et bande il est conseillé de n'y toucher qu'en dernier ressort, après avoir épuisé les autres possibilités (modifications des réglages de volume et de tonalité).

En principe ces réglages seront positionnés ainsi :

— tonalité presque au maximum de graves,

— volume à moitié.

Noter que le circuit de sauvegarde et de chargement sur cassettes est prévu pour fonctionner avec un magnétophone équipé d'une prise « LINE » aux normes DIN, mais que la machine fonctionne parfaitement sur une entrée « MICRO » et une sortie « écouteur » moyennant le respect des conseils ci-dessus.

Les cassettes : il est inutile d'utiliser des cassettes de très haute qualité (chrome ou métal) le magnétophone à cassettes utilisé ne permettant pas, à moins qu'il ne soit de « haut de gamme » de profiter de leur qualité. On choisira des cassettes fiables mécaniquement et d'une durée maximum de 60 minutes.

Il existe des cassettes de courte durée (5, 10, 15 minutes par face) que nous conseillons d'utiliser pour plusieurs raisons :

— la faible quantité de bande qu'elles contiennent permet un défilement très régulier,

— on accèdera plus rapidement aux différents programmes d'où un gain de temps appréciable lors du chargement, surtout si le magnétophone est muni d'un compteur.

DERNIERS CONSEILS

— Nettoyer fréquemment les têtes magnétiques à l'aide d'un coton tige imbibé d'alcool à 90°. Le galet et l'axe cabestan seront aussi nettoyés par ce procédé.

— Démagnétiser de temps en temps les têtes et parties métalliques du magnétophone à l'aide d'un démagnétiseur ou d'une cassette de démagnétisation. Cette opération, trop souvent négligée, rendra un air de jeunesse à votre appareil.

— Doubler toujours les sauvegardes de programmes sur deux cassettes différentes. L'une sera (si possible de type courte durée) enregistrée en vitesse rapide (2 400 bauds), l'autre, du type C 60 par exemple, contiendra les programmes en vitesse lente (suffixe S lors de la sauvegarde, vitesse 300 bauds).

— Protéger la cassette de sauvegarde en brisant les languettes qui interdisent l'enregistrement. Ces languettes peuvent toujours être remplacées par un petit morceau d'adhésif.

— Séparer les divers enregistrements par un banc de quelques secondes où vous enregistrerez, grâce au micro, le titre du programme qui suit, si votre mégnétophone n'est pas équipé d'un compteur.

— Noter titres et repères compteurs, sur la jaquette de la cassette. Moyennant toutes ces précautions, il est peu probable que vous rencontriez des problèmes de cassettes.

Le téléviseur : il est évident que, si vous désirez utiliser au mieux les possibilités de l'ORIC, vous choisirez un modèle couleur. Il devra être muni d'une prise PERITEL, sinon vous devrez vous procurer un codeur/modulateur, qui permettra, en partant des signaux RVB et SYNCHRO émis par l'ordinateur, de les coder en SECAM et de moduler avec le signal résultant un oscillateur UHF dont le signal sera introduit sur l'entrée correspondante

de téléviseur. La qualité sera moins bonne qu'avec une prise PERITEL, l'avantage sera l'utilisation possible d'un téléviseur ancien non équipé de la dite prise.

Pour nos applications en communication, la couleur n'est pas indispensable et un téléviseur noir et blanc reste utilisable. Hélas, il devra être du type multistandard car le signal issu du modulateur UHF de l'ORIC n'est pas aux normes françaises.

Pour réutiliser un téléviseur standard français, deux solutions vont se présenter :

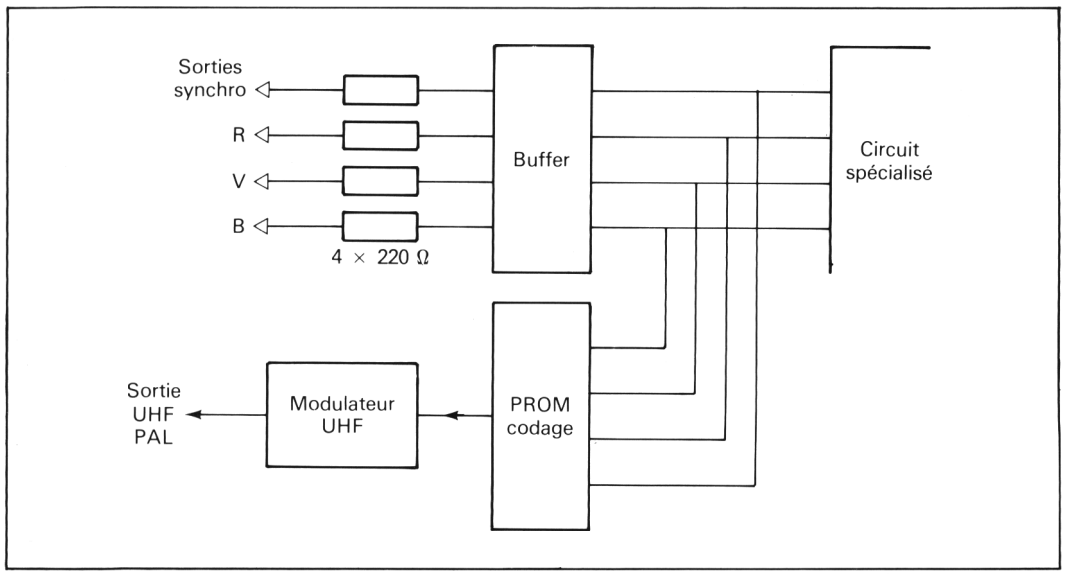
— accès direct à l'ampli vidéo du téléviseur (moniteur).

— adjontion d'un modulateur UHF extérieur à l'ORIC.

Dans ces deux cas, nous devrons câbler une prise, sur la sortie « RVB » de l'ORIC comme suit. Les points 1, 2, 3 et 4 de la dite prise seront ensemble. L'âme d'un câble coaxial miniature y sera connectée, la tresse soudée au point 5 de la prise.

On a « mélangé » les signaux R, V, B et synchro de l'ORIC, les combinant en un signal unique « vidéo composite ». Ceci est rendu possible de par la composition interne des circuits de l'ORIC.

L'autre extrémité du câble coaxial pourra être connectée à l'entrée vidéo du monteur ou du téléviseur. C'est aussi avec ce signal que l'on pourra moduler l'oscillateur UHF extérieur pour attaquer le tuner du téléviseur ou l'émetteur de télévision d'amateur qui pourra ainsi transmettre des textes ou dessins de l'ORIC. Le niveau du signal disponible sous 75 Ω est d'environ 800 mV.



SYNOPTIQUE DES SORTIES RVB/SYNCHRO ET UHF PAL DE L'ORIC

IMPORTANT : Nous tenons à signaler que l'utilisation de l'ORIC sur une entrée vidéo (avec une liaison par câble coaxial, comme décrite ci-dessus), a résolu pratiquement tous les problèmes d'interférences avec la station radio en réception. Cette solution est donc séduisante pour l'utilisation en CW et RTTY et à chaque fois que la couleur n'est pas nécessaire.

Nous n'avons pas essayé d'effectuer un blindage complet et efficace du câble de liaison PERITEL-ORIC qui offrirait peut-être la même solution au brouillage, avec l'avantage de pouvoir conserver la couleur.

STRUCTURE DE L'ORIC-1

Nous allons déshabiller l'ORIC pour examiner ses entrailles. Ceci nous permettra par la suite de tirer le meilleur parti de cette structure, lors de la conception de nos programmes.

Pour notre voyage dans l'ORIC, nous vous conseillons de suivre sur le synoptique général puis d'examiner les parties plus détaillées.

L'ORIC est organisé autour d'un microprocesseur 8 bits du type 6502A. Son horloge est à 1 MHz. Nous verrons plus en sétail le fonctionnement de ce composant dans un chapitre qui sera consacré à la programmation en langage machine et en assembleur.

Un autre composant essentiel est l'U.L.A., spécialement développé pour l'ORIC, circuit à haute densité d'intégration, dont la tâche est principalement orientée vers la gestion d'écran, mais qui participe également à d'autres menus travaux.

Les entrées-sorties sont assurées par un VIA, type 6522, qui s'occupe du générateur sonore, de l'imprimante, en partie du clavier en collaboration avec le port intégré au générateur sonore, et des opérations cassette et imprimante.

Le générateur sonore AY 8912 attaque un petit ampli BF intégré, de type LM 386 qui excite le haut-parleur de l'ORIC. La programmation du VIA et du générateur sonore, étant possible à l'utilisateur averti, nous consacrerons un chapitre complet à ces deux composants.

Le signal présent sur la bande magnétique est amplifié, filtré et mis en forme par un double amplificateur opérationnel LM 358.

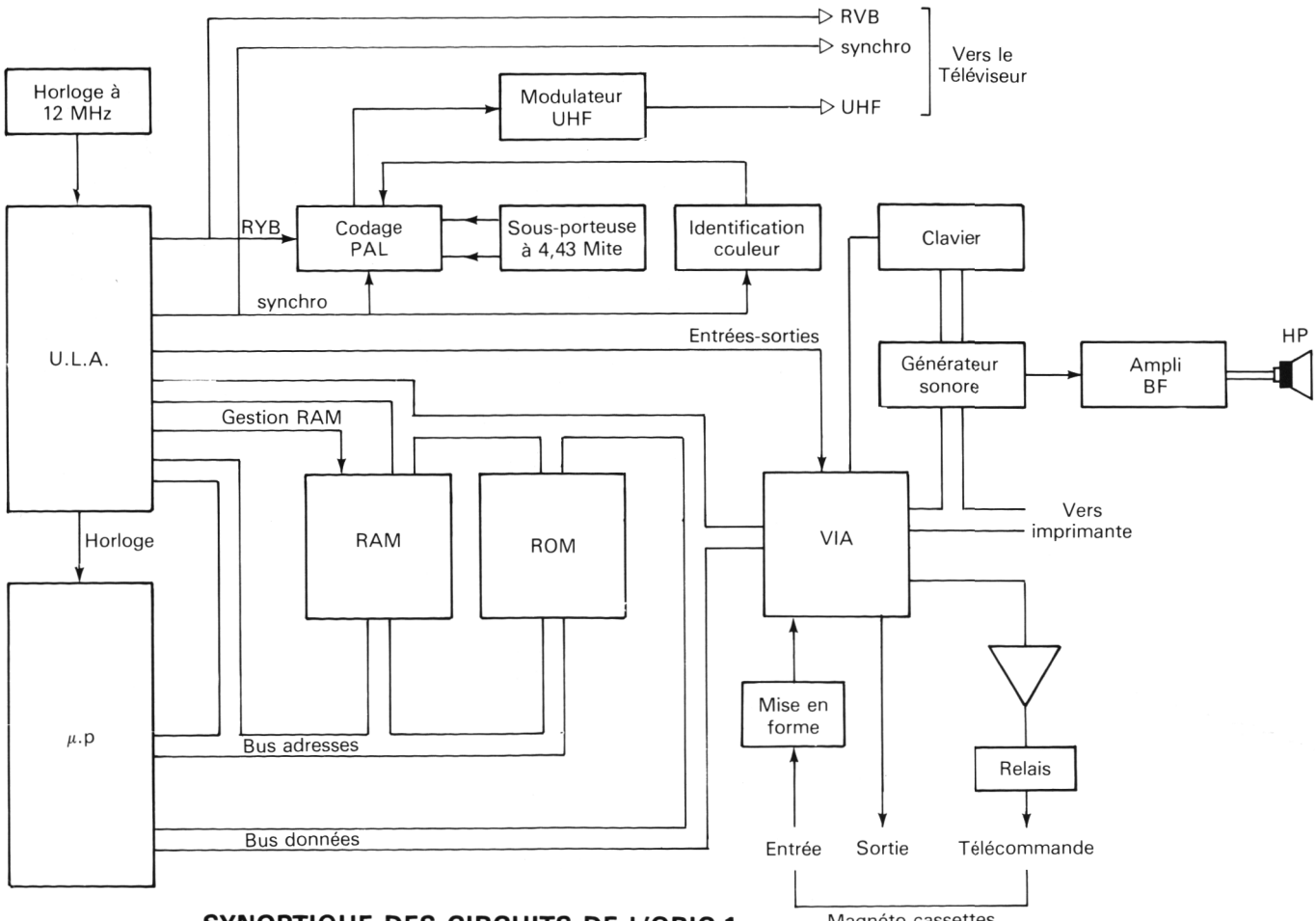
Nous n'avons pas encore parlé des circuits envoyant le signal vers le téléviseur. Nous savons déjà que le travail de gestion de l'écran est accompli par l'U.L.A. qui génère aussi les signaux RVB et synchro.

Ces signaux sont codés pour former le signal PAL par une PROM. La sous-porteuse à 4,43 MHz est fournie par un oscillateur piloté par un quartz de 8,867 MHz. Les « burst » couleur sont fabriqués à partir du signal de synchronisation par un monostable.

Toutes les informations sont contenues dans une mémoire vive constituée par des RAM dynamiques.

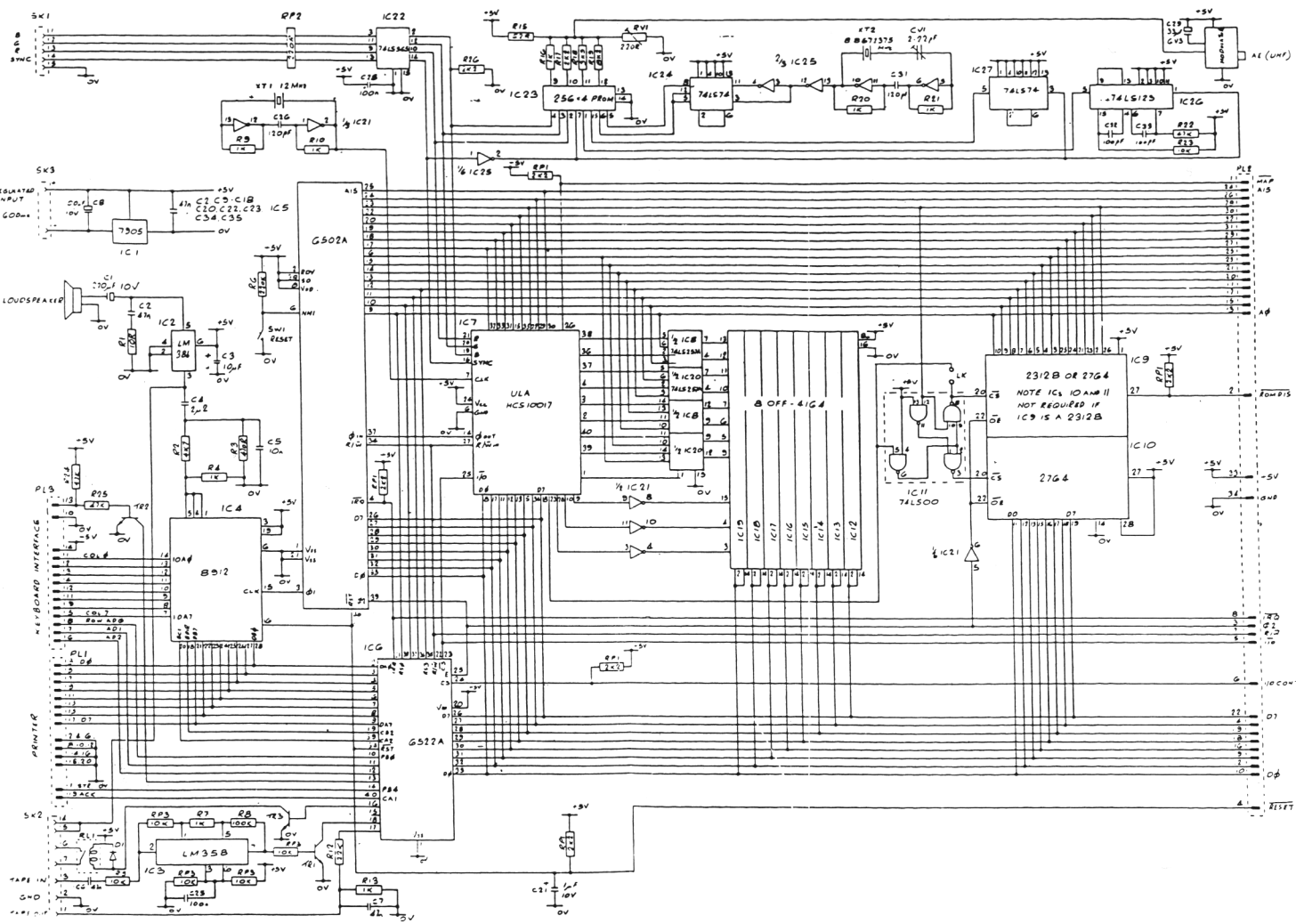
Le chef d'orchestre de toute cette circuiterie est la mémoire ROM qui contient le langage BASIC, implanté sur 16 K Octets. Cette mémoire peut, selon les différents modèles d'ORIC, être constituée soit d'une ROM masquée soit d'une EPROM (2 × 2 764).

Tout le système utilise des circuits logiques, hormis l'ampli BF et le circuit de mise en forme des signaux lus par la cassette, qui sont du type analogique et dont nous fournissons les schémas.



SYNOPTIQUE DES CIRCUITS DE L'ORIC-1

Schéma électrique de l'ORIC-1

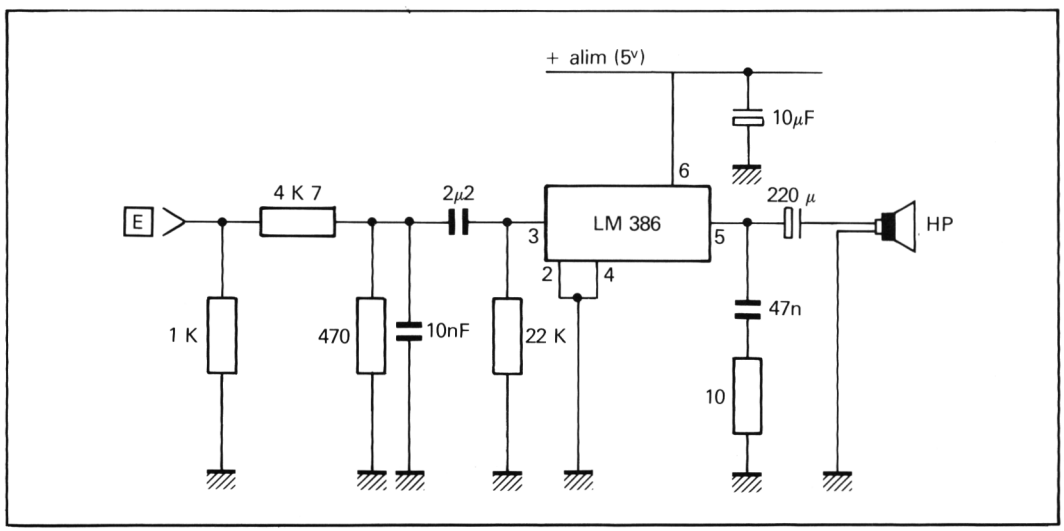


Le montage est réalisé sur un circuit imprimé unique, double face d'aspect soigné, sur lesquel vient se relier le clavier par l'intermédiaire d'un connecteur.

Les bus adresses et données du micro-processeur ainsi que les signaux de contrôle (notamment la gestion des entrées-sorties) sont disponibles sur le connecteur d'extention de l'appareil.

On regrettera cependant qu'il soit nécessaire d'utiliser une alimentation supplémentaire pour la prise PERITEL...

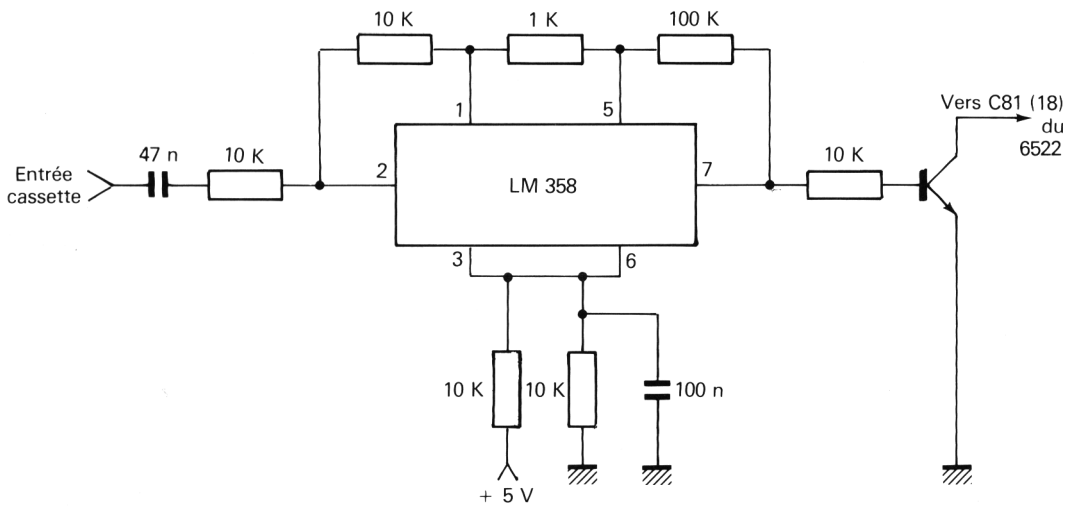
AMPLI BF DE L'ORIC



Le point E est relié aux 3 sorties (3 voies) du circuit générateur sonore (AY 8912). Les 3 voies se trouvent ainsi « mélangées ». C'est le point 3 du LM 386 qui est relié à la prise de sortie vers un ampli extérieur.

Si vous utilisez cette sortie vers l'ampli extérieur, nous vous conseillons de mettre un condensateur en série entre le point 3, entrée du LM 386, et l'extérieur, faute de quoi un court-circuit avec la masse ou une tension accidentelle trop élevée peut être fatale au LM 386.

CIRCUIT DE LECTURE DE LA CASSETTE



Le signal prélevé sur la sortie cassette est filtré puis amplifié et rendu compatible TTL, avant d'être envoyé à la broche CB 1 du VIA 6522 qui le traitera.

POSSIBILITÉS D'EXTENSION

Nous allons apprendre à utiliser au mieux les composants internes de l'ORIC pour éviter de devoir disposer de circuits externes trop nombreux. Néanmoins il est impossible de tout faire sans extension externe. C'est la raison pour laquelle le constructeur a prévu une zone d'adresses réservées aux composants externes.

Ces composants adressés par l'ORIC seront commandés par, et fourniront des signaux de contrôle. Ce sont ces signaux que l'on retrouve sur le connecteur d'extension de l'ORIC aux points suivants :

En 1 : **MAP** signal d'entrée permet d'inhiber la ROM (quand la RAM est adressée) ou la RAM (quand la ROM est adressée). On pourra ainsi connecter aux bus des ROM ou RAM externes.

En 6 : **I/O CONTROL** est aussi un signal entrant, qui doit être fourni par un périphérique externe ayant été adressé. Ce signal va alors inhiber le VIA interne à l'ORIC.

En 5 : **I/O** est un signal de sortie indiquant qu'une adresse d'entrée-sortie est présente sur le bus.

Le manuel de programmation de l'ORIC nous apprend que les adresses de la page 3, de 0300 à 03FF sont réservées aux Entrées-Sorties.

En utilisant judicieusement cette page mémoire et les signaux de contrôle pré-cités, nous pourrons accroître les possibilités d'Entrée-Sortie de la machine.

PARTAGE DE LA MÉMOIRE DE L'ORIC

La ROM, la RAM et les entrées-sorties se partagent l'espace mémoire adressable (64 K octets).

La zone occupée par la ROM est immuable et s'étend de FFFF à C000. Une autre zone n'est pas modifiée, c'est celle qui occupe les 5 premières pages de la mémoire.

En page zéro (0 à 00FF) on trouvera des variables système, entre 0100 et 01FF la pile de travail, de 0200 à 02FF différentes variables, entre 0300 et 03FF les organes d'entrée-sortie et de 0400 à 04FF les adresses de sous-programmes.

Le programme commencera en 0500 et s'étendra jusqu'à 97FF (ou jusqu'à B3FF si on utilise la fonction GRAB).

Les adresses sont différentes, ensuite selon le mode de fonctionnement choisi, HIRES ou TEXT.

1er clavier 9800 à 9BFF (HIRES) B400 à B7FF (TEXT).
2e clavier 9C00 à 9FFF (HIRES) B800 à BB7F (TEXT).
ÉCRAN A000 à BFDF (HIRES) BB80 à BFDF (TEXT).

Nous verrons au cours de la lecture de cet ouvrage que certaines de ces adresses sont importantes ou utiles.

DE L'ORIC-1 VERS L'ATMOS

Mis à part les différences extérieures d'aspect physique, dues au nouveau clavier, ORIC-1 et ATMOS sont strictement identiques électroniquement parlant. Tout ce qui fait la différence, c'est la ROM des deux machines.

Les BASIC sont identiques et occupent 16 K de mémoire. En grande partie, ils sont compatibles entre eux ce qui autorise le chargement d'un programme ORIC-1 sur ATMOS mais, nous le verrons, il n'y aura pas de lancement automatique et la machine vous affichera ERRORS FOUND. En fait, il n'y a pas lieu de s'alarmer sur ce point puisque ce message d'erreur a tendance à apparaître de façon un peu aléatoire sur ATMOS, ce qui semble dû à une petite erreur dans le sous-programme qui effectue le chargement et la vérification d'un programme cassette. Ce qu'il faut faire dans ce cas, avant le RUN, c'est vérifier par un LIST que le programme qui apparaît à l'écran n'est pas « pollué » (caractères parasites, zones d'écriture en vidéo inversée, numéros de lignes fantaisistes ou chargement incomplet — suites de « U » —). En parcourant le listing, on recherchera également la présence de POKE à des adresses concernant les variables de gestion du système (page ∅ et page 2 de la mémoire), qu'il nous faudra interpréter par la suite, pour savoir s'il y a lieu, ou non, de les modifier. On aura tout intérêt à noter sur une feuille de papier, les numéros de lignes correspondants. Il faudra également prendre note de tous les CALL faisant appel à une adresse en ROM (# C∅∅∅ à # FFFF). Attention, dans ce cas les opérations sont plus délicates à effectuer, car un CALL peut appeler un sous-programme en RAM qui utilise lui-même un sous-programme en ROM...

Il n'y a pas de recette miracle : ou le programme fonctionne correctement quand on fait RUN (après les modifications « évidentes »), ou bien il se plante à cause de l'appel, par une routine en langage machine, d'un sous-programme en ROM : il faudra désassembler pour effectuer toutes les modifications. Ce sera le cas le plus difficile à traiter mais l'opération reste possible avec beaucoup de patience.

Pour mener à bien ces différents travaux, nous allons examiner les diverses adresses des variables et routines essentielles des deux machines.

STOUT F82F (ATMOS) F865 (ORIC-1)
Affichage d'un texte sur la ligne supérieure de l'écran.

Conditions	A	octet de poids faible de l'adresse du texte
	Y	octet de poids fort de l'adresse du texte
	X	position horizontale (0 à 39) début écriture
Retour	X	contient la prochaine position d'écriture.

Le message doit se terminer par ∅∅

VDU F77C (ATMOS) F73F (ORIC-1)
Affichage d'un caractère (ou d'un caractère de contrôle) à la prochaine position de l'écran, et déplace le curseur.
Conditions X contient le code ASCII du caractère
Retour pas de registre modifié.

GTORK EB78 (ATMOS) E905 (ORIC-1)
Saisie d'un caractère à partir du clavier.
Le code ASCII se trouve, au retour de la routine, placé dans A.

PRTCHR F5C1 (ATMOS) F57B (ORIC-1)
Permet d'envoyer des caractères à l'imprimante.
Conditions A contient le code ASCII du caractère
Retour le registre A est modifié.

ROUTINES GRAPHIQUES ET SONORES

Sur ATMOS et ORIC-1, les routines graphiques ou sonores s'utilisent de la même façon.

Il existe, à partir de l'adresse de 2E∅ une zone tampon où sont rangés les différents paramètres à transmettre à ces routines. Ces paramètres sont codés sur deux octets, en complément à 2. L'octet situé en 2E∅ est un flag d'erreur. Avant d'entrer dans une routine, il y a lieu de le positionner à ∅. Il sera mis à 1 si la routine utilisatrice détecte une erreur (paramètre de valeur interdite). En principe les contenus des registres A, X, Y seront modifiés ce qui implique leur sauvegarde indispensable avant l'accès aux routines.

Nous présentons ci-dessous, sous forme de tableau les rôles des différents paramètres pour chaque routine. C'est l'adresse de poids faible du paramètre qui est portée dans le tableau. Pour un paramètre codé sur un seul octet, il y aura lieu de prendre la précaution de forcer à zéro son octet de poids fort.

Les valeurs X et Y représentent, par les routines graphiques, les coordonnées. La valeur FB a le même sens que dans l'instruction BASIC (autorisés : 0, 1, 2, 3).

TABLEAU DES ROUTINES ET PARAMETRES GRAPHIQUES

ROUTINE	ORIC-1	ATMOS	2E1	2E3	2E5	
CURSET	F∅2D	F∅C8	X	Y	FB	
CURMON	F∅64	F∅FD	X	Y	FB	
DRAW	F∅79	F11∅	X	Y	FB	
CHAR	F∅A5	F12D	code ASCII	4^3 standard 1 graphique	FB	
CIRCLE	F2E5	F36F	Rayon	FB		
PATRN	F∅93	F11D	profil			
POINT	F∅3C	F1C8	X	Y		Au retour 2E1
FILL	F1E5	F268	Nb de colonnes	Nb de cellules	Valeur	contient : ∅ si arrière plan 1 si avant plan
PAPER	F17F	F2∅4	couleur			(couleur)
INK	F18B	F210	couleur			

TABLEAU DES ROUTINES ET PARAMETRES SONORES

ROUTINE	ORIC-1	ATMOS	2E1	2E3	2E5	2E7
SOUND	FB26	FB4∅	Canal	Période	Volume	
MUSIC	FBFE	FC18	Canal	Octave	Note	Vol.
PLAY	FBB6	FBD∅	Tone	Bruit	Mode enve-loppe	Pé-riode enve-loppe

TABLEAU DES FONCTIONS SONORES PRE-PROGRAMMEES

	ORIC-C	ATMOS
PING	F412	FA9F
SHOOT	F415	FAB5
EXPLODE	F418	FACB
ZAP	F41B	FAE1
BIP clavier	FAFA	FB14
BIP CTRL	FB1∅	FB2A

W8912

Est une routine qui permet d'accéder à la programmation du AY-3-8912, le générateur sonore.

La donnée contenue dans le registre X est envoyée au registre du 8912 dont le numéro est spécifié par A.

Conditions A n° du registre du 8912 donnée à envoyer
Retour Rien.

Nous avons donc fait le tour des routines essentielles qui sont souvent utilisées dans les programmes en langage machine.

Nous aurions pu encore citer :

* **F89B** qui régénère sur ORIC-1 la forme des caractères du clavier d'origine après leur modification et dont l'adresse est F8D∅ sur ATMOS.
* **F430** ORIC-1 **F8B2** ATMOS ont le même effet qu'un appui sur RESET.
* **CC**∅A ORIC-1 **CCCE** ATMOS exécuteront un effacement de l'écran (CLS).
* **E6CA/E8**∅4 sur ORIC sont **E76A/E93D** sur ATMOS et font l'inhibition et l'autorisation clavier.
* **C5F8** ORIC-1 **C5E8** ATMOS permettent la saisie d'un caractère au clavier, dont on récupère le code dans A.
* **CBED** de l'ORIC-1 est en **CCB**∅ sur ATMOS avec les mêmes conditions d'accès (provoque affichage d'un texte).

VARIABLES DE GESTION DU SYSTÈME

Sont inchangées, les variables de la page zéro 9A à A7 qui fournissent les adresses des débuts ou fin de zone programme et variable.

En page 2, on retrouve au même endroit et pour le même rôle :

208 et # 209 qui renseignent sur la touche pressée

26A variable étant des différentes bascules

26B et # 26C qui renseignent sur les couleurs PAPER et INK.

Par contre le nombre de lignes sur l'écran (affectées par le CLS ou le Scrolling) est défini sur ATMOS par :

27E (au lieu de # 26F) pour le nombre de lignes

27C et # 27D qui indiquent le nombre de caractères à scroller. Ainsi, pour afficher sur une fenêtre écran réduite, il faudra modifier ces 3 paramètres.

Exemple : on ne veut utiliser que les 10 lignes supérieures en mode 40 colonnes ; on devra :

1 — passer en 40 colonnes (# 26A)

2 — mettre 10 en # 27E (POKE # 27E, 10)

3 — mettre 10 × 40 = 400 en # 27C (DOKE # 27C, 400).

Nous ne citons ici que quelques variables couramment utilisées.

Pour l'utilisation du magnétophone, il faudra se souvenir que les variables de la page zéro (# 5F ~ # 64) sont remplacées par les adresses # 2A9 à # 2AE. C'est # 24D qui gère le SLOW-FAST au lieu de # 67.

En résumé, nous dirons que :

Les programmes ORIC-1 pourront en général être transformés rapidement pour une utilisation sur ATMOS. Pour optimiser cette transformation on devra les reprendre en détail pour supprimer les artifices de programmation employés pour masquer les « bogues » de la ROM d'ORIC-1. Nous ne citerons que pour mémoire le trop fameux TAB qui demandait l'addition d'un offset de 13. Laissé tel quel, tous les PRINT seront décalés sur ATMOS. La suppression du premier octet, par traitement de chaîne (code de fonction) l'ors de l'utilisation de STR $ / PLOT est un autre exemple.

Enfin, il faudra tirer parti des nouvelles fonctions (STORE et RECALL) de l'ATMOS remplacées sur ORIC-1 par des routines en langage machine, de manipulation de données.

CODAGE ET IMPLANTATION D'UN PROGRAMME EN MÉMOIRE

Nous avons vu que, sur ORIC, le programme BASIC commençait en mémoire à l'adresse 0501. Il est donc aisé de chercher à comprendre comment est organisée, codée et implantée en mémoire, une ligne de programme. Allant plus loin, nous chercherons à voir où se trouvent les variables et leur mode de représentation. Il peut être utile, dans certains programmes de pouvoir sauvegarder ces seules zones de mémoire, moyennant il est vrai, la modification de quelques pointeurs.

Pour ce faire, nous avons écrit un petit programme, sans aucune signification réelle, et au moyen d'un programme de lecture de mémoire, qui lui est écrit en langage machine, nous avons lu chaque adresse mémoire. Après cet examen, on peut comprendre le principe adopté sur ORIC pour coder les programmes.

La machine doit être capable de faire la différence entre ce qui est programme et... les variables. Les limites entre ces zones de mémoires sont connues grâce à des pointeurs.

Ces pointeurs sont, pour la plupart, localisés dans la page zéro de la mémoire. Un poin teur contient une adresse. Il est donc représenté sur deux octets. L'octet codant le poids faible de l'adresse est situé le premier, celui qui représente le poids fort arrive ensuite.

Les pointeurs que nous allons examiner dans ce chapitre sont au nombre de quatre :

— Le premier, indique l'adresse de début du programme BASIC. C'est l'adresse à partir de laquelle viennent se ranger les octets lus sur la bande magnétique ou saisis au clavier.

Ce pointeur est situé en 9A. Il contient l'adresse 0501.

— Le second pointeur qui retiendra notre attention, indique le début de la zone des variables. Sa situation en page zéro est à l'adresse 9C.

— Le troisième pointeur intéressant se trouve à l'adresse 9E. Il indique l'implantation mémoire des tableaux de variables et de chaînes de caractères.

— Le dernier pointeur est situé en A0. Il contient l'adresse de fin de zone des variables de toutes sortes.

LA COMPOSITION D'UNE LIGNE DE PROGRAMME

Pour bien comprendre ce paragraphe, veuillez vous reporter aux listings présentés.
La ligne : 10 DIM A (5)
se traduit en mémoire de la manière suivante :

Adresse	Octet lu	
0501	0B	Représentation de l'adresse mémoire du début de la
0502	05	ligne suivante (050B)
0503	0A	Représentation du numéro de la ligne
0504	00	000A = ligne 10
0505	93	Code l'ordre DIM
0506	41	Code ASCII lettre A
0507	28	Code ASCII parenthèse gauche
0508	35	Code ASCII chiffre 5
0509	29	Code ASCII parenthèse droite
050A	00	Terminateur de ligne.

Chaque ligne commence donc par deux octets servant à indiquer où se trouve en mémoire la ligne suivante. Le numéro de la ligne présente est codé sur deux octets également.

Les ordres BASIC sont représentés sur un seul octet, ce qui permet déjà, en premier lieu, un léger gain de place dans la mémoire de programme. Tous ces codes sont supérieurs à 7F. En effet, les valeurs inférieures servent au code ASCII de représentation des signes et lettres, majuscules et minuscules.

Quand l'interpréteur BASIC arrive sur l'octet correspondant à un « ordre », il le décode grâce à une table localisée dans la ROM. C'est la forme littérale de cet ordre qui sera, bien évidemment affichée à l'écran.

Les caractères suivants sont représentés par leur code ASCII.

La ligne est close par un zéro. Il y a lieu d'insister sur un point remarquable. Si vous avez eu la curiosité d'interrompre le chargement d'un programme (appui sur touche PAUSE ou STOP du magnétophone suivi d'une action sur le poussoir RESET) vous avez sûrement remarqué, en demandant un listing que celui-ci apparaissait à l'écran... jusqu'à l'endroit où la lecture du programme avait été interrompue. Chaque ligne est donc considérée comme un bloc indivisible qui est matérialisé sur la bande magnétique.

En poursuivant ainsi notre lecture mémoire, nous arrivons sur la dernière ligne et constatons que l'adresse de branchement correspondante nous envoie sur deux octets à zéro : ils matérialisent la fin de la zone programme. On trouve trois zéro consécutifs : un qui appartient à la dernière ligne, deux pour la fin du programme.

LISTINGS POUR EXAMEN ZONE MÉMOIRE

```
0500 00 0B 05 0A 00 93 41 28 ......A(
0508 35 29 00 17 05 14 00 93 5)......
0510 42 43 24 28 33 29 00 23 BC$(3).#
0518 05 1E 00 41 28 30 29 D4 ...A(0).
0520 31 32 00 2F 05 23 00 41 12./.#.A
0528 28 31 29 D4 32 34 00 40 (1).24.@
0530 05 28 00 41 28 32 29 D4 .(.A(2).
0538 31 32 33 34 2E 35 36 00 1234.56.
0540 4C 05 2D 00 41 28 33 29 L.-.A(3)
0548 D4 31 35 00 5A 05 2F 00 .15.Z./.
0550 43 4E D4 CD 31 2E 33 35 CN..1.35
0558 31 00 6D 05 32 00 42 43 1.m.2.BC
0560 24 28 31 29 D4 22 46 36 $(1)."F6
0568 47 4B 51 22 00 80 05 37 GKQ"...7
0570 00 42 43 24 28 32 29 D4 .BC$(2).
0578 22 46 31 45 5A 48 22 00 "F1EZH".
0580 8F 05 3C 00 53 24 D4 22 ..<.S$."
0588 53 41 4C 55 54 22 00 A0 SALUT"..
0590 05 46 00 BA 22 4F 4B 22 .F.."OK"
0598 3B 42 43 24 28 31 29 00 ;BC$(1).
05A0 BA 05 50 00 BA E7 28 23 ..P...(#
05A8 39 43 29 2C E7 28 23 39 9C),.(#9
05B0 45 29 2C E7 28 23 41 30 E),.(#A0
05B8 29 00 00 00 00 00 00 00 ).......
05C0 00 00 00 00 00 00 00 00 ........
05C8 00 00 00 00 00 00 00 00 ........
05D0 00 00 6E 00 B9 49 2C 30 ..n..I,0
05D8 00 26 05 78 00 90 00 00 .&.x....
05E0 00 43 4E 81 AC ED 91 69 .CN....i
05E8 53 80 05 88 05 00 00 41 S......A
05F0 00 25 00 01 00 06 84 40 .%.....@
05F8 00 00 00 85 40 00 00 00 ....@...
0600 8B 1A 51 EB 85 84 70 00 ..Q...p.
0608 00 00 00 00 00 00 00 00 ........
0610 00 00 00 00 42 C3 13 00 ....B...
0618 01 00 04 00 00 00 05 66 .......f
0620 05 05 79 05 00 00 00 55
```

ZONE MÉMOIRE CORRESPONDANTE

PROGRAMME UTILISÉ

```
10 DIMA(5)
20 DIMBC$(3)
30 A(0)=12
35 A(1)=24
40 A(2)=1234.56
45 A(3)=15
47 CN=-1.351
50 BC$(1)="F6GKQ"
55 BC$(2)="F1EZH"
60 S$="SALUT"
70 PRINT"OK";BC$(1)
80 PRINTDEEK(#9C),DEEK(#9E),DEEK(#A0)
```

LA ZONE DES VARIABLES

Après la zone programme, que trouve-t-on ? Voyons le listing. 43, 4E, 81, AC...

Tiens ! 43 est le code ASCII de la lettre C.

4E est celui de la lettre N.

C'est la variable CN que nous avions définie à la ligne 47, suivie d'une valeur. Evidemment, ce qui suit est moins simple car il ne s'agit ni plus ni moins que de la représentation en virgule flottante de la valeur attribuée à la variable. Ce codage est effectué sur 5 octets : 1 pour l'exposant et 4 pour la mantisse.

Poursuivons...

53 est le code ASCII de la lettre S.

80 semble ne rien représenter... c'est en fait l'octet qui aurait servi à coder la deuxième lettre de la variable... si elle avait eu deux lettres (en effet ORIC n'effectue sa reconnaissance que sur les deux premières lettres). Pourquoi 80 ? En fait, c'est la valeur 00 avec le bit de poits fort à 1 d'où 80. Dans la représentation des chaînes de caractères, le premier bit du second octet du nom est forcé à 1. La valeur 05 correspond à la longueur de la chaîne S $ (5 caractères). Cette valeur est suivie de deux octets : 88,05. Ils représentent l'adresse d'initialisation de la variable dans le programme (0588 ce que l'on peut vérifier sur le listing).

Les 2 octets suivants sont à zéro : ils sont inutilisés.

En allant plus loin, on trouve le tableau de variables A que nous avions DIMmensionné à 5 au début du programme.

Représentation d'un tableau : (tableau A).

41 00 codent la lettre A (2 octets sont réservés)

25 00 constituent la valeur à ajouter à l'adresse mémoire où se situe le tableau, pour passer au suivant. Ici 5CA + 25 = 5EF. Suivent le nombre de dimensions du tableau et la représentation en virgule flottante des différents éléments.

Représentation d'un tableau de chaînes (tableau BC$)

42 C3 codent le nom B = 42 et C = 43 mais avec le 1er bit forcé à 1 43 devient C3.

13 00 représentent la position de la variable suivante, suivent le nombre de dimensions, le nombre d'éléments, puis chaque élément, précédé de sa longueur et désigné par son emplacement mémoire dans le programme.

Voici en gros, le schéma de la répartition de la mémoire entre le programme et les variables. Il est possible de tirer parti de cette connaissance dans certains cas, en sauvegardant par exemple les valeurs de variables.

Pour illustrer ce chapitre, nous vous proposons un petit programme simple de renumérotation. Ce programme ne renumérote pas les GOTO ni les GOSUB mais il présente l'intérêt d'être court. On peut donc l'introduire rapidement à la suite d'un programme quand on s'est aperçu que la numérotation est trop serrée pour ajouter des lignes supplémentaires. Dans ce cas, il y aura lieu de noter tous les branchements avant la renumérotation, pour les refaire ensuite.

Le principe de ce programme sera simple. Sachant que le début du programme est en 0501, il suffira d'adopter le principe suivant :

1 — Lecture du pointeur adresse (celui qui donne l'adresse en mémoire de la ligne suivante).

```
63950 REM ++++    RENUM    ++++
63951 STOP
63955 CLS
63957 INPUT"NUMERO 1ERE LIGNE ";DEP
63958 INPUT"PAS DE LA RENUM   ";PAS
63959 PTR=#501
63962 REPEAT
63965 DOKE(PTR+2),DEP
63968 DEP=DEP+PAS
63971 PTR=DEEK(PTR)
63975 UNTILDEEK(PTR+2)>=63950
63999 END
```

2 — Ecriture du nouveau numéro de la première ligne.

3 — Incrémentation (en fonction du pas choisi) du numéro de ligne et dépôt de ce nouveau numéro à l'adresse indiquée par le pointeur.

4 — On recommence jusqu'à la fin du programme.

Comme le sous-programme de renumération réside en mémoire derrière le programme l'utilisant, il ne faut pas qu'il se renumérote lui-même. C'est le rôle du test à la ligne 63975.

On lance la renumérotation par RUN 63955.

Le programme utilisant la RENUM ne devra bien sûr pas avoir de ligne de numéro supérieur à 63950. Il vous appartiendra de renuméroter les GOTO, GOSUB et autres branchements calculés.

DERNIERS POINTS SUR L'EXAMEN DE LA MÉMOIRE

Si vous lisez la mémoire par une boucle FOR-NEXT contenant un PRINT PEEK, vous finirez par trouver, après la zone contenant le programme BASIC et celle où sont situées toutes les variables (ordinaires ou chaines), un endroit à partir duquel on ne lit que la valeur 85 ou # 55 (ou le caractère U, si vous avez fait PRINT CHR $...).

Cette valeur correspond à celle qui est écrite en RAM, permettant le test de celle-ci, lors de la mise sous tension de la machine. Elle est le profil binaire 01010101 (# 55) où un bit sur 2 est à 1, l'autre étant à zéro.

Vous pourriez pousser plus en avant cet examen de la RAM. Il est possible de constater que, lors du mauvais chargement d'un programme on obtient souvent, en demandant le listing, un numéro de ligne erroné ou un caractère « parasite » au sein d'une ligne.

En déterminant l'adresse dans la ligne « polluée », du caractère ou du numéro de ligne indésirable, il est presque toujours possible de s'en sortir en le corrigeant par POKE. (On

détermine son adresse par une boucle FOR-NEXT faisant apparaître les caractères par PRINT CHR $...). Ceci ne vaut que pour des listings très faiblement « pollués ».

Une instruction BASIC pourra même être modifiée de cette manière puisqu'elle est codée sur un seul octet. Ce code est un nombre compris entre 128 et 248.

INTRODUCTION AU LANGAGE

- NOTIONS DE PROGRAMME ET DE LANGAGE
- LE BASIC DE L'ORIC PARTICULARITÉS DÉFAUTS
- VARIABLES DE GESTION DU SYSTÈME
- LA ROM

NOTIONS DE PROGRAMME ET DE LANGAGE

Le lien qui existe entre l'opérateur et la machine, au travers du clavier et de l'écran de visualisation, est le langage. Sans la compréhension de ce langage, la machine ne serait qu'un ensemble de circuits intégrés inertes. Cette compréhension, nous dirons interprétation, du langage est assurée, sous le contrôle du micro-processeur, par la ROM interne.

Le micro-processeur ne peut travailler qu'en langage binaire. Il serait horriblement long et fastidieux d'introduire les programmes dans sa mémoire, sous ce format. Le rôle de langage sera donc d'établir le lien entre l'opérateur et le micro-processeur.

NOTION DE PROGRAMME

Avant d'en arriver au langage et puisque nous avons parlé de binaire, nous introduirons la notion de programme comme étant une suite d'instructions. L'instruction est d'élément de base compris par la logique du micro-processeur. Comme les instructions ne peuvent parvenir n'importe comment au microprocesseur, il faut les organiser en une suite ordonnée qui sera le reflet de l'action que l'opérateur désire faire exécuter à la machine. Cette suite ordonnée d'instructions est le programme.

LE LANGAGE

Le programme est écrit par l'opérateur dans un certain langage capable d'être compris par la machine. Pour éviter à l'opérateur d'écrire en binaire, on a créé des langages « évolués ».

Ces langages sont plus abordables à l'homme et sont traduits ensuite dans la machine. Pour être compris correctement, le langage doit répondre à certains critères. Ce sont un peu les règles de grammaire qui régissent les langues.

En effet, on imagine mal une langue qui ne serait composée que de mots de vocabulaire qu'on pourrait assembler n'importe comment. Le sens de la phrase, à supposer que celle-ci demeure compréhensible, en serait tout changé.

F6GKQ appelle F1EZH pourrait indifféremment devenir
F1EZH appelle F6GKQ ou
F1EZH F6GKQ appelle ou
F6GKQ F1EZH appelle ce qui n'est pas toujours pareil !

Ces règles de grammaire sont appelées la SYNTAXE de la phrase de programmation.

Cette syntaxe devra être respectée et, l'un des rôles de « l'analyseur » qui reçoit les phrases que vous introduisez dans la machine, sera de contrôler cette syntaxe avant exécution du programme et de signaler les erreurs.

NOTION DE COMPILATEUR ET D'INTERPRÉTATION

La traduction du langage peut être effectuée de deux manières différentes.

Compilation : le programme est entièrement traduit en langage machine avant de pouvoir être exécuté, et cela, le plus souvent en deux lectures ou « passes ». Il y a transformation de langage SOURCE en langage OBJET. C'est la compilation. Le programme qui réalise cette opération est appelé COMPILATEUR. La compilation n'a lieu qu'une seule fois.

Interprétation : le programme est analysé, ligne par ligne, et chaque ligne est exécutée. A chaque lancement de programme il y a une nouvelle interprétation de chaque ligne avant son exécution. Ce travail est réalisé par un INTERPRÉTEUR.

La première méthode est beaucoup plus efficace et on trouve, sur certains micro-ordinateurs de haut de gamme des « BASIC » compilés ou semi-compilés. Les programmes sont exécutés beaucoup plus rapidement, la compilation n'ayant lieu qu'une fois pour toutes.

Le BASIC interprêté permet néanmoins l'utilisation de la machine en mode conversationnel, par l'intermédiaire de ses organes d'entrée-sortie (clavier, écran, imprimante...). Il est « interactif ».

La machine, nous le savons, peut être utilisée pour exécuter des ordres immédiats, comme une calculatrice.

PROGRAMMER, c'est donc connaître à la fois l'éventail des instructions acceptées par la machine, et qui constituent en quelque sorte son vocabulaire, et leur syntaxe d'utilisation. Mais programmer, c'est aussi savoir définir l'ordre d'enchaînement de ces instructions, pour arriver à un programme performant et en évitant de ré-écrire plusieurs fois des tâches répétitives. Le regroupement de ces tâches répétitives dans une structure de « sous-programme » rendra plus clair le programme principal.

L'ANALYSE ET L'ALGORITHME

Pour parvenir à une programmation efficace d'un problème donné, il faut avoir procédé auparavant à l'analyse détaillée de ce problème. Connaissant bien les fins, on déterminera, au mieux, les moyens.

La phase finale de l'analyse doit se conclure par l'élaboration d'un organigramme ou algorithme, faisant apparaître clairement les tâches à réaliser.

L'écriture du programme au clavier ne devra commencer qu'après l'élaboration de cet algorithme.

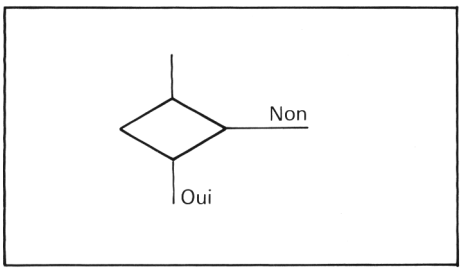
Plusieurs algorithmes différents peuvent conduire à la résolution d'un même problème, parfois avec la même efficacité. On pourra mettre l'accent sur la rapidité ou sur la concision ou encore sur la clarté du programme résultant.

Dans tous les cas, il y aura lieu de commenter abondamment l'algorithme et aussi le programme (instructions REM), ce qui devra permettre à un utilisateur futur de pouvoir introduire d'éventuelles modifications. On néglige trop souvent l'écriture de la documentation associée à un programme en pensant gagner du temps. En fait, l'expérience montre que, souvent, ce temps est perdu par la suite, si des modifications doivent avoir lieu.

Exemple d'algorithme : la table de multiplication. Soit à programmer la table de multiplication par 2 entre 1 et 10.

Les tâches seront : 1 — initialiser le multiplicande,
2 — initialiser le multiplicateur,
3 — effectuer la multiplication et imprimer résultat,
4 — incrémenter le multiplicateur et répéter l'opération 3 jusqu'à atteindre la limite fixée (10).

L'algorithme pourra s'écrire ainsi, en convenant que le petit losange placé sous un rectangle indique un test (oui ou non, GO ou NO-GO pour les anglais) et que la ligne partant du losange à l'horizontale correspond à la blranche NON. On pourra ainsi omettre les inscriptions OUI et NON à chaque test.



Le programme résultat pourrait s'écrire :

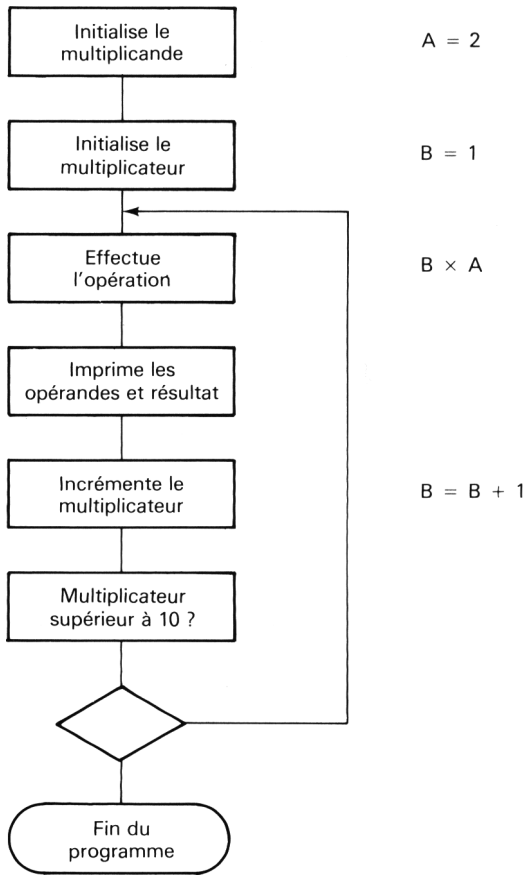
```
10    A = 2
20    B = 1
30    PRINT A ; « * » ; B ; « = » ; A * B
40    B = B + 1
50    IF B < 10 THEN GOTO 30
60    END
```

Mais on utilisera bien sûr de préférence la structure de boucle FOR-NEXT permise par BASIC.

```
10    A = 2
20    FOR B = 1 TO 10
30    PRINT A ; « * » ; B ; « = » ; A * B
40    NEXT B
```

Tout cela peut paraître bien simple ici, et reconnaissons-le, pas vraiment indispensable pour un tel programme, l'opération d'analyse étant réalisée « de tête », mais sur un programme complexe, il ne faudra pas négliger cette phase.

Organigramme de la table de multiplication



LE BASIC DE L'ORIC

Le BASIC de l'ORIC est très standard, fort proche des versions MICROSOFT très répandues. Notre propos n'est pas de le décrire en détail et nous conseillons à nos lecteurs intéressés, de lire les ouvrages spécialisés consacrés à ce langage. Nous nous attarderons seulement sur ses particularités pour tenter de ne pas tomber dans un certain nombre de pièges.

Un bref passage en revue des instructions nous permettra de les classer par familles.

1 — MATHÉMATIQUES

ABS, ATN, COS, EXP, INT, LN, LOG, PI, RND, SGN, SIN, SQR, TAN

Regrettons l'absence des fonctions trigo inverses. Le manuel de programmation nous indique comment les retrouver. Nous avons eu besoin de ces fonctions dans le programme de calcul de distance (QRA-Locator).

La fonction **RND** ne fournit pas réellement un nombre aléatoire. Ce défaut est surtout évident sur un programme qui se lance automatiquement, après la mise sous tension de la machine. La séquence générée est toujours la même. On pourra la remplacer avantageusement en lisant une suite d'octets dans la ROM ou en consultant par exemple l'octet de poids faible du compteur du VIA (voir le chapitre des « variables système »).

2 — FONCTIONS LOGIQUES

AND, OR, NOT, FALSE, TRUE

Les opérateurs logiques seront surtout utilisés dans les tests de condition **(IF THEN ELSE)**. Noter que, tout comme, en langage machine, on peut les utiliser pour faire un masque de bits **(AND)**.

FALSE et **TRUE** sont deux variables booléennes qui n'existent pas sur tous les BASIC.

3 — FONCTIONS D'ÉDITION

L'éditeur est la partie du langage qui permet de travailler avec l'écran. C'est lui qui gère l'introduction des lignes nouvelles, l'affichage, les déplacements du curseur etc...

Il faut reconnaître que l'éditeur de l'ORIC est bien peu pratique, tant au niveau du nombre de caractères admis sur une même ligne de programme, qu'à celui des modifications que l'on désire apporter à cette ligne.

Le manuel reste assez succinct sur ce sujet. Notons qu'il est néanmoins possible d'ajouter des caractères au milieu d'une ligne, sans avoir à l'écrire de nouveau. On utilisera la propriété suivante : les mouvements du curseur commandés par les flèches ne modifient pas une ligne de programme.

Supposons qu'après avoir écrit la ligne suivante :

50 PRINT « BONSOIR, A DEMAIN »

vous désiriez la changer par :

50 PRINT « BONSOIR ET A DEMAIN ».

L'introduction du mot ET et la suppression de la virgule s'effectuent grâce à la procédure suivante. (Si elle est au milieu d'un programme, il peut être pratique de l'isoler auparavant, au bas de l'écran, par EDIT 50.)

- Déplacer le curseur par **CTRL A** jusqu'au blanc qui suit le R de BONSOIR.
- Sauter la virgule en bougeant le curseur par → (flèche droite).
- Descendre le curseur par ↓ (flèche bas).
- Ecrire les deux lettres ET.
- Repositionner à l'aide des flèches ↑ et ←, le curseur devant l'espace précédant le A.
- Valider alors le reste de la suite par **CTRL A** puis **RETURN.**

Outre les mouvements de curseur et les caractères de contrôle **(CTRL)** notons les commandes

EDIT, LIST (et LLIST), TRON, TROFF

Les ordres **TRON** et **TROFF** ne sont pas présents dans tous les BASIC et sont bien pratiques pour rechercher d'éventuelles erreurs.

4 — MANIPULATIONS MÉMOIRE

CLEAR, NEW, FRE, GRAB, RELEASE, PEEK, POKE, DEEK, DOKE

FRE est utilisable pour se débarrasser de variables inutiles, notamment les copies de chaînes, mais aussi pour connaître à tout instant la quantité de mémoire qui reste disponible.

GRAB et **RELEASE** n'existent pas sur tous les BASIC. utiles pour les longs programmes elles permettent l'utilisation d'une zone mémoire supplémentaire (9800 à B400).

Si **PEEK** et **POKE** sont classiques et permettent de lire et d'écrire directement en mémoire, **DEEK** et **DOKE** sont moins communes et sont très pratiques car elles permettent de lire (ou d'écrire) une information codée sur 2 octets (notamment les adresses...).'

Attention, un défaut de l'ORIC, **POKE** suivi d'une valeur en hexadécimal ne marche pas toujours...

5 — LECTURE ET INITIALISATIONS DE VALEURS

LET, DIM, READ / DATA, RESTORE

Peu de particularités. Regrettons seulement que **RESTORE** n'agit qu'en réinitialisant le pointeur **DATA** en première ligne. C'est un peu dommage et il faudra triturer les variables sytème pour qu'il en soit autrement.

6 — TRAITEMENT DES CHAINES DE CARACTÈRES

ASC CHR$, HEX$, LEFT$, RIGHT$, MID$, LEN, STR$, VAL

Toutes ces fonctions sont classiques et se retrouvent sur les autres BASIC standards.

A cause des tares cachées de l'ORIC, **STR$, VAL, LEN** risquent de vous réserver quelques surprises.

7 — LES BRANCHEMENTS

GOTO (et ON... GOTO), GOSUB (et ON... GOSUB) RETURN, CALL permettent le déroutement des programmes **CALL** autorise l'accès à un sous-programme écrit en langage machine.

IF THEN ELSE complètent la panoplie, introduisant la notion de condition.

8 — ÉCRITURE, TRACÉS À L'ÉCRAN

CLS, CHAR, CIRCLE, CURMOV, CURSET, DRAW, FILL, PATTERN, PLOT, POINT, PRINT, SCRN SPC, TAB

Le propos de cet ouvrage n'est pas, nous le rappelons, d'entrer dans les détails du BASIC, aussi nous vous renvoyons à la notice de la machine. Regrettons simplement l'absence du « PRINT USING » permettant de formatter des impressions et le défaut énorme sur **TAB** (voir paragraphe « tares du BASIC »).

Signalons que **PRINT** peut être abrégé « ? ».

9 — LES STRUCTURES DE BOUCLES

FOR-NEXT (STEP), REPEAT-UNTIL

Le **REPEAT-UNTIL** est pratique en ce sens qu'il permet d'exécuter une tâche répétitive jusqu'à la réalisation d'une condition prédéfinie. La sortie d'une boucle **FOR-NEXT** autrement que sur sa limite supérieure risquant de poser des problèmes on utilisera la structure **REPEAT UNTIL,** plus adaptée.

10 — SAISIE DE DONNÉS

GET, INPUT, KEY$
Classiques...

11 — COMMANDES SONORES

PLAY, SOUND, MUSIC commandent le générateur sonore de façon standard.
EXPLODE, SHOOT, PING, ZAP génèrent des bruits pré-programmés.

12 — GESTION COULEUR ET RÉSOLUTION

INK, PAPER gèrent les couleurs de l'écriture et du fond.
TEXT, LORES, HIRES déterminent la résolution graphique.

13 — FONCTIONS DIVERSES

STOP-CONT, RUN-END, DEF FN-DEF USR, CLOAD-CSAVE, REM
Ces fonctions ont une action identique à celles des autres BASIC.
WAIT permet de générer un délai multiple de 10 ms.
POP-PULL agissent sur la pile des adresses de retour des sous-programmes et seront familières aux utilisateurs du langage machine, qui les trouveront adaptées au BASIC.

En résumé nous dirons que le BASIC ORIC est rapide (pour un langage interprété), complet et standard. Cette dernière qualité permet une adaptation aisée des programmes prévus pour d'autres machines.

Nous lui reprochons l'absence de sauvegarde de données et fichiers ce qui obligera l'utilisateur à concevoir des sous-programmes de remplacement pour pallier ce défaut.

Nous serons moins tendres envers certaines bogues énormes qui entâchent la ROM contenant le BASIC VERSION 1.0. Nous vous livrons ces erreurs pour éviter qu'elles ne conduisent le lecteur débutant sur la machine à des recherches inutiles.

LES TARES DU BASIC

L'écriture d'un programme peut réserver quelques surprises.

a) Le **TAB** (du **PRINT TAB**) ne fonctionne que si on ajoute 13 à la valeur de tabulation désirée. De plus, il fonctionne par rapport au dernier **PRINT** réalisé ce qui est très gênant.

b) **STR$** ajoute un caractère devant la variable transformée en chaîne. Ainsi :

```
AN = 1984
AN$ = STR$ (AN)
PRINT AN$ donne bien 1984 mais
PRINT LEN (AN$) donne... 5 !
```

PLOT 10, 10, AN$ écrit 1984 en VERT ce qui est parfois illisible sur un téléviseur Noir et Blanc.

PRINT VAL (AN$) donne 0...

Pour supprimer ce caractère parasite qui vient se mettre devant la chaîne AN $, il faut traiter cette dernière au moyen de la fonction MID $ par exemple. On écrira : AN$ = MID$ (AN$, 2).

c) **LLIST** permettant de sortir vers l'imprimante pose aussi quelques difficultés : lorsqu'on regarde le listing de près on constate qu'il manque des caractères. Ce défaut est très gênant si le listing doit servir de témoin à un autre programmeur.

Ce problème est lié au temps perdu par ORIC à la lecture de son clavier. On peut le résoudre en faisant :

POKE # 307, 255 : LLIST ou encore CALL # EDO1 : LLIST

ce qui a pour effet de modifier le temps de scrutation clavier. Une action sur RESET remettra la bonne valeur dans le compteur du VIA ou réautorisera la lecture clavier.

d) **POKE**. L'écriture en mémoire par l'instruction POKE est à manier avec précautions. En effet :

POKE 48003, # 41 donne une erreur de syntaxe. Il faut écrire POKE 48003, 65 pour qu'ORIC accepte la commande (provoque affichage d'un A en 4e colonne de la ligne supérieure de l'écran).

L'adresse du **POKE** peut être écrite en hexadécimal, mais la valeur doit l'être en décimal. Ainsi POKE # BB83, 65 est accepté.

Notons que le problème n'existe pas avec **DOKE**.

DOKE 48003, # 41 fonctionne.

Signalons enfin que, la notice signale page 23, une manipulation « en général » plus rapide des variables entières (celles qui sont précédées du signe %) que des variables en virgule flottante. Il semblerait que ce soit plutôt le contraire...

```
10    B = 3
20    FOR I = 1 TO 5000
30    B = B + 1
40    NEXT
50    PING
```

L'exécution de ce programme demande 28 secondes. Si on remplace B par B% il faut 33 secondes avant le gong !

Ainsi, si vous recherchez la rapidité d'exécution, préférez des variables à une seule lettre (aux variables dites « entières » %). L'avantage des variables entières reste le gain de place en mémoire.

D'autres points semblent flous. Ainsi sur les 2 machines différentes utilisées par les auteurs de ce livre,

```
10    GET A$
20    PRINT ASC (A$)
```

donne pour l'apostrophe (') illegal quantity error sur une machine et 255 pour l'autre machine.

Rappelons que le résultat correct serait 39.

Pour en déterminer avec ces critiques sur le contenu de la ROM, signalons que les observations ont été faites sur des machines équipées du BASIC V1.0. Ces deux machines présentent déjà des différences... Les programmes présentés dans ce livre fonctionnent pourtant parfaitement sur les deux.

La partie matérielle de la machine peut aussi vous perturber. Si le générateur sonore tombe en panne dans une certaine configuration, comme il intègre un port utilisé par le clavier, vous perdez le contrôle de celui-ci. Dans ce cas, il est parfois possible de continuer à utiliser la machine en utilisant la commande **CTRL F** qui inhibe le clavier sonore. Cette astuce n'est hélas pas valable dans tous les cas...

LES VARIABLES DE GESTION DU SYSTÈME

Nous n'avons pas d'informations sur ces paramètres importants, dans la documentation de base fournie par le constructeur. Il faut donc effectuer bon nombre de recherches personnelles et faire des recoupements avec les résultats obtenus par d'autres amateurs pour arriver à obtenir les adresses de ces quelques variables.

Le meilleur moyen permettant d'atteindre ce but est l'exploration de la ROM, en supposant au départ que les variables sont situées entre les pages 0 et 4 de la RAM.

Nous avons vu dans un chapitre précédent comment s'organisait le partage et l'implantation mémoire entre programme et variables. Les variables système suivantes ont été mises à jour, occupant chacune deux octets :

9A : adresse de début du programme
9C : adresse de fin du programme
9E : adresse de fin des variables
A0 : adresse de fin des tableaux

Nous nous bornerons à ne citer que les plus utiles et vous retrouverez la plupart d'entre elles dans les programmes proposés dans cet ouvrage.

1 — NUMÉRO DE LIGNE, NUMÉRO DE COLONNE

La position d'écriture à l'écran est définie, en mode texte par deux variables contenant le numéro de la ligne (# 268) et le numéro de la colonne (# 269). Ces numéros vont de 0 à 26 pour les lignes (27 lignes) et de 0 à 39 pour les colonnes (ceci est important quand on place ORIC en 40 colonnes).

Nous pouvons dès lors simuler le **PRINT AT** qui est absent du BASIC ORIC. Examinons le programme ci-dessous

```
 5  CLS
10  POKE # 268, 9          prépare position écriture en
20  PRINT                    10e ligne
30  POKE # 269, 15         écrit la valeur de PI en 10e ligne, 16e colonne
40  PRINT
```

Auparavant vous aurez fait **CTRL**] pour passer en 40 colonnes. Cela fonctionne également en mode 38 colonnes (TEXT normal) mais il faut se souvenir alors que les 2 premières colonnes sont utilisées pour les codes couleur.

2 — NOMBRE DE LIGNES ÉCRAN

Le nombre de lignes utilisées pour l'affichage est contrôlé par une variable située en # 26F. Si vous la modifiez, vous pouvez partager votre écran en 2 zones par exemple. La partie inférieure de l'écran sera protégée d'un **CLS**. Les textes qui y seront inscrits ne pourront plus être effacés, jusqu'à réinitialisation de la valeur de la variable.

```
 5 CLS
20 PLOT 1, 30, « ------ .... ----- »
30 PLOT 13, 22, « PROTÈGE »
40 POKE # 26F, 20
50 CLS
```

Vous constatez que le second **CLS** n'efface pas le bas de l'écran.

Si vous possédez un téléviseur couleurs, ajoutez au programme ci-dessus, après avoir actionné **RESET**,

```
10 PAPER 5
45 POKE # 26B, 18
```

Après le **RUN**, l'écran est mauve en bas et vert en haut. En # 26B on contrôle la couleur de la fenêtre écran.

Pour en finir avec les fonctions d'impression, on peut connaître la prochaine position d'impression, ou la forcer, grâce à l'adresse # 12 qui contient une adresse entre BB00 et BFDE.

3 — GESTION DU MAGNÉTOPHONE

L'essentiel des variables de gestion des sauvegardes et lectures sur cassettes est situé en page zéro.

La routine de sauvegarde les utilise et on a besoin de savoir :

a) la vitesse de l'opération,

b) si le programme est en machine ou BASIC,

c) si l'exécution doit être automatique,

d) les adresses de début et de fin de programme, pour la sauvegarde des blocs de mémoire,

e) le titre, mais il est facultatif...

Le relais de commande du moteur est piloté par un bit du VIA : il s'agit du bit 6 du port B (nous verrons, plus en détail les constitutions et programmations du VIA). En écrivant le bon profil binaire dans le registre correspondant, on commandera le relais. En fait, on agit non pas sur l'état de la ligne PB6 pour le faire, mais sur sa commande de direction (registre DDRB). Si on ne veut pas modifier les autres bits du registre, il faut confectionner un masque (opérateur logique **AND**).

Dans le programme RTTY les valeurs utilisées pour la commande relais émission/réception (adresse 0302) les valeurs sont :

Relais collé : 167 = # A7 = 10101111
Relais décollé : 231 = # E7 = 11101111

Les autres adresses citées plus haut sont :

# 67	vitesse	0	pour 300	1 pour 2 400 bds
# 64	type	0	BASIC	1 machine

# 63	lancement	0 par RUN	1 auto	
# 62 # 61	adresse fin de zone mémoire à sauvegarder.			
# 60 # 5F	adresse début de zone mémoire à sauvegarder.			

Entre # 35 et # 46 on peut loger un titre.

4 — PARAMÈTRES DIVERS

Une autre variable très intéressante contrôle divers paramètres. Située à l'adresse # 26A elle simule en permanence l'effet que vous pourriez obtenir par les caractères de controle (CTRL).

Les bits suivants sont utilisés :

LSB	0	CURSEUR	(CTRL Q)	0 sans	1 avec
	1	VIDEO	(CTRL S)	0 sans	1 avec
	3	SON CLAVIER	(CTRL F)	0 avec	1 sans
	4	ESCAPE	(CTRL [)	0 sans	1 avec
	5	COLONNES	(CTRL])	0 38	1 40
	6	HAUTEUR	(CTRL D)	0 simple	1 double

En écrivant le profit correct en 26A par un POKE vous obtiendrez les options de votre choix. Les bits 2 et 7 sont inutilisés.

5 — CHRONOMÈTRE

Une horloge est disponible aux adresses # 276 (et # 277). Cette horloge vous est accessible. Vous pouvez la lire par **DEEK** ou la modifier par **DOKE**. Attention toutefois à son utilisation en langage machine car si vous avez inhibé les interruptions (SEI) elle n'est plus utilisable. Cette horloge est décrémentée toutes les 10 ms. On peut aussi utiliser le compteur de temps du VIA mais il y a lieu de noter qu'il varie beaucoup plus rapidement, au rythme de l'horloge micro-processeur (décrémenté toutes les micro-secondes). Nous retrouverons ce compteur lors de l'étude du VIA.

6 — LE CODAGE DU CLAVIER

Une valeur, fonction de la touche clavier actionnée est disponible à l'adresse # 208. Si aucune touche n'est préssée, on obtient # 38. Cette adresse pourra avantageusement être utilisée dans les routines en langage machine. Les touches CTRL et SHIFT sont codées en # 209 selon le même principe.

Le code ASCII des touches est disponible en # 35.

Nous n'avons ni cité, ni exploré, toutes les variables système mais seulement celles qui nous apparaissent comme étant les plus utiles, ou les plus faciles à interpréter.

Dans un même état d'esprit, nous allons fournir quelques adresses de routines en ROM, que vous pourrez atteindre depuis un programme écrit en machine.

VALEURS DE L'OCTET # 208 ET TOUCHES CORRESPONDANTES

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	7	J	M	K	Sp.	U	Y	8	N	T	6	9	,	I	H	L
9	5	R	B	;	.	O	G	0	V	F	4	—	↑	P	E	/
A									1	Esc	Z		←	Del	A	Ret
B	X	Q	2	\	↓	]	S		3	□	C	'	→	[	W	=

TABLEAU DES TOUCHES ET CODE HEXA ATTRIBUÉ

A	AE	N	88	0	97	ESC	A9
B	92	O	95	1	A8	→	BC
C	BA	P	9D	2	B2	←	AC
D	B9	Q	B1	3	B8	↑	96
E	9E	R	91	4	9A	↓	B4
F	99	S	B6	5	90	.	94
G	96	T	89	6	8A	,	8C
H	8E	U	85	7	80	'	BB
I	8D	V	98	8	87	;	93
J	81	W	BE	9	8B	—	9B
K	83	X	B0	SPACE	84	/	9F
L	8F	Y	86	DEL	AD	\	B3
M	82	Z	AA	RETURN	AF	=	BF
		[	BD	]	B5		

LES BONNES ADRESSES DE LA ROM

Parvenus à ce stade, il y a lieu d'insister (au risque de se répéter), sur un point bien précis : toutes les adresses que nous avons citées dans les chapitres précédents et celles que nous allons livrer, correspondent à la ROM BASIC version 1.0. Toute autre version risque d'entraîner bien des changements.

La programmation en langage machine est, nous le verrons, très efficace et absolument nécessaire dans certains cas. Il peut néanmoins être très utile de ne pas ré-écrire des routines qui existent déjà dans la ROM.

ACCÈS À L'ÉCRAN

a) La routine d'impression d'un caractère à l'écran est localisée en # F73F. Pour y accéder, il suffit de charger le code ASCII du caractère à imprimer dans le registre X et d'appeler la routine.

Les caractères de mouvement de curseur (retour-arrière, saut de ligne, effacement écran) sont admis.

La routine type sera donc

```
LDX  %     # 41
JSR        F73F
(RTS)
```

pour écrire la lettre A à l'écran.

Les autres registres ne sont pas modifiés.

b) L'effacement de l'écran peut être obtenu en mettant dans X la valeur # 0C et en appelant la routine en F73F, mais aussi et plus simplement en écrivant

```
JSR     # CC0A
(RTS)
```

LECTURE DU CLAVIER

La saisie d'un caractère à partir du clavier est effectuée grâce à la routine située en # C5F8. Au retour de cette routine, le code ASCII du caractère correspondant à la touche est dans l'accumulateur (A). Il peut être mémorisé, traité ou affiché...

Le schéma type d'une saisie de clavier avec affichage du caractère correspondant sera donc le suivant :

```
JSR     # C5F8
TAX
JSR     # F73F
(RTS)
```

Le caractère présent, dans A après saisie du clavier, est transféré dans X avant l'appel à la routine d'affichage.

INHIBITION DU CLAVIER

En # E6CA se trouve une routine qui a pour effet d'inhiber la lecture du clavier. Cela peut-être utile pour accélérer l'exécution d'un programme ou tout simplement pour interdire toute action de l'utilisateur.

La routine en # E804 remettra les choses en ordre et réautorisera la lecture du clavier.

ED01 provoque aussi l'inhibition clavier (pour LLIST par exemple).

ACCÈS AU GÉNÉRATEUR SONORE

Un chapitre de ce livre est consacré au générateur sonore (AY-8912) et à sa programmation. La routine d'accès est en # F535. Le registre à modifier doit être dans A et la valeur dans X.

```
LDA  %     # Numéro registre
LDX  %     # Valeur à écrire
JSR     # F535
(RTS)
```

LE MAGNÉTOPHONE

Après avoir initialisé les diverses variables de la page zéro comme expliqué dans le chapitre « variables système », il est possible d'accéder à la routine de sauvegarde sur cassette située en # E57B.

De même, la lecture se fera grâce à la routine ROM en # E4A8.

Il faut néanmoins penser à inhiber le clavier. (# E6CA).

ÉCRITURE SUR LIGNE SUPÉRIEURE DE L'ÉCRAN

Une routine située en # F82F permet d'aller écrire sur la ligne supérieure de l'écran. Cela peut-être utile pour faire apparaître un commentaire, qui ne sera pas effacé par CLS, par exemple. Le texte à écrire est en mémoire. Il doit se terminer par « 00 ». Bien sûr, il ne peut excéder 40 caractères, terminateur compris...

L'accumulateur doit contenir l'octet de poids faible de l'adresse d'implantation du message, le registre Y l'octet de poids fort, le registre X le numéro de colonne où doit commencer l'affichage. On aura donc :

```
LDA  %  # 00
LDY  %  # 09
LDX  %  # 05
JSR  #  F82F
(RTS)
```

Dans cet exemple le message est supposé être implanté en # 0400 et sera écrit à la 6e colonne. Essayez le programme ci-dessous

```
10 AD=#0400
20 READD$
30 IFD$="999"THEN1000
40 D=VAL("#"+D$)
50 POKEAD,D:AD=AD+1:GOTO20
100 DATA46,36,47,4B,51,00
110 DATAA9,00,A0,04,A2,05,20,2F,F8
120 DATA60,999
1000 CALL#0406
```

ÉCRITURE D'UN TEXTE

```
10 AD=#0400
20 READD$:IFD$="999"THEN1000
30 D=VAL("#"+D$):POKEAD,D:AD=AD+1:GOTO20
100 DATA42,4F,4E,4A,4F,55,52,0A,0D,0A,46
    ,36,47,4B,51,0A,0A,0A,00,EA ,EA,EA,EA

110 DATAA9,00,A0,04,20,ED,CB,60
120 DATA999
1000 CLS:CALL#0417
```

LISTING DU PROGRAMME DÉSASSEMBLÉ

```
7F
0417 A900    LDA %#00
0419 A004    LDY %#04
041B 20EDCB JSR #CBED
041E 60      RTS
```

ZONE MÉMOIRE DE RANGEMENT DU TEXTE

```
06

0400 42 4F 4E 4A 4F 55 52 0A BONJOUR.
0408 0D 0A 46 36 47 4B 51 0A ..F6GKQ.
0410 0A 0A 00 EA EA EA
```

ÉCRITURE D'UN TEXTE SUR L'ÉCRAN

En # CBED de la ROM, se situe une routine qui permet d'afficher un texte sur l'écran. Ce texte sera rangé dans une zone mémoire déterminée, dont on fournira l'adresse de début à la routine d'exécution. L'accumulateur contiendra l'octet de poids faible de l'adresse, le registre Y, l'octet de poids fort.

Le texte devra obligatoirement se terminer par 00. Il pourra contenir des caractères tels que SAUT DE LIGNE, mouvements curseur, commandes couleurs etc...

Un court programme de démonstration est fourni ici. Le texte est implanté à partir de # 0400. Le programme l'utilisant est en # 0417.

Toutes ces routines pouvant être appelées par un programme utilisateur, écrit en langage machine, vous seront d'une grande utilité pour raccourcir l'écriture de vos programmes.

DEUX CIRCUITS UTILES DE L'ORIC

- LE VIA
- LE CIRCUIT SONORE

LA STRUCTURE ET LA PROGRAMMATION DU VIA

Nous avons vu que le VIA 6522 était un des composants essentiels de l'ORIC. Nous ne détaillerons pas entièrement sa structure et sa programmation, car vous pouvez trouver ces informations dans la fiche technique complète du constructeur du composant, mais notre propos vise à vous fournir suffisamment d'éléments pour que vous puissiez l'utiliser correctement.

La structure d'ORIC utilise le VIA pour les commandes :

- Imprimante
- Clavier
- Générateur sonore
- Magnétophone à cassettes.

Nous pouvons modifier dans une certaine mesure l'attribution de ces fonctions à condition de reprogrammer le VIA.

STRUCTURE DU VIA

VIA signifie en fait Versatile Interface Adapter. C'est un circuit d'interfaçage d'emploi « souple » ou « universel ».

Pour communiquer avec ses différents périphériques, le micro-processeur, et dans notre cas, le micro-ordinateur a besoin d'un circuit d'interface permettant de rendre compatibles entre eux différents signaux.

La commande de relais, la lecture de contacts etc... passent par des lignes qui sont regroupées par 8. On les appelle PORTS.

Le VIA contient 2 PORTS d'entrée-sortie qu'on appellera PORT A et PORT B. Ce sont donc 16 lignes d'Entrée-Sortie qui seront programmables indépendamment les unes des autres dans le sens Entrée ou Sortie. A ces PORTS sont associées des lignes de contrôle ou d'indication.

Ces lignes sont les mêmes que celles qu'on retrouve dans les PIA ou PIO, mais le VIA est plus complexe car il possède en outre :

2 compteurs-timer.

1 registre à décalage.

Tout ce petit monde peut fonctionner sous le contrôle d'une logique d'interruption.

Le fonctionnement général du composant est régit par l'intermédiaire de 16 registres. Quatre lignes du bus adresse sont donc cablées sur notre VIA. Des signaux de sélection viennent le commander.

LES PORTS A ET B

La programmation d'un PORT passe par un registre particulier, appelé registre de direction (DDRA ou DDRB) qui fixe le sens de chaque ligne du PORT. Pour programmer une ligne en sortie, on écrira un 1 ; en entrée un 0 sur le bit correspondant. Ainsi :

11001010 = # CA = 202 mettra les lignes 7, 6; 3, 1 en sortie, les autres en entrée.

Quand le sens de la ligne a été fixé en sortie, son état haut ou bas est déterminé par les bits du registre de sortie du PORT correspondant (ORA et ORB). Ainsi, faisant suite à la programmation ci-dessus :

01000010 = # 42 = 66 mettra les lignes 7 à l'état bas, 6 haut, 3 bas, 1 haut.

Les adresses des registres sont :

0300 ORB
0301 ORA
0302 DDRB
0303 DDRA

L'écriture en assembleur des exemples ci-dessus sera (PORT B)

```
LDA  % # CA
STA  # 0302
LDA  % # 42
STA  # 0300
```

A ces PORTS sont associées des lignes de contrôle dont la programmation est suffisamment complexe pour ne pas la détailler ici. Ces lignes sont CA1, CA2 pour le PORT 1, CB1, CB2 pour le PORT B.

Elles peuvent fonctionner en entrée ou sortie. Ce mode est régit par la programmation du registre PCR, registre de commandes périphériques.

Les bits 7, 6, 5 sont consacrés à CB2
Le bit 4 est affecté à CB1
Les bits 3, 2, 1 sont consacrés à CA2
Le bit 0 à CA1

Ainsi pour les demandes d'interruption sur CA1 et CB1 on aura un 1 dans le bit correspondant, pour une demande sur un front montant du signal surveillé, un 0 sur une transition négative.

Le registre PCR est à l'adresse 030C.

Quand le sens d'utilisation du signal est déterminé, on peut tester les bits correspondants du registre IFR, registre indicateur d'interruptions. Les états dans ce registre sont mémorisés. Il est à l'adresse 030D. L'affectation des bits du registre IFR est la suivante :

7 IRQ
6 T1
5 T2
4 CB1
3 CB2
2 SR (registre à décalage)

1 CA1
0 CA2

Le bit correspondant est à zéro. Il passe à 1 si un signal, de sens défini par la programmation de PCR est détecté.

Le bit 7 est à 1 dès qu'un bit du registre est à 1. La surveillance du registre iFR peut donc se faire par scrutation périodique du Bit 7. Après lecture, il faut penser à remettre le bit à zéro pour une utilisation future. Cette remise à zéro s'effectue en écrivant 1 à l'emplacement du bit. Ainsi :

```
00000001 = 01     LDA % # 01
                  STA   # 030D
```

met à zéro l'indicateur d'interruption de CA2

```
11111111 = # FF   LDA % # FF
                  STA   # 030D
```

met tous les indicateurs à zéro.

Le registre activateur d'interruptions (IER) est à l'adresse 030E. Il possède la même affectation de bits que le registre IFR. Un bit à 1 autorisera l'interruption correspondante. Un bit à 0 va l'interdire.

Le bit 7 du registre IER à un rôle bien spécial. Si on le force à 0, chaque 1 du profil binaire envoyé à IER mettra à zéro le bit correspondant. Si on le force à 1, chaque 1 va autoriser l'interruption correspondante.

Ainsi on aura :

```
LDA % # 1E (00011110)
STA # 030F
```

pour mettre à zéro les bits 6, 6 et 0 (T1, T2 et CA2).

Le registre de commande auxiliaire (ACR) implanté en mémoire à l'adresse 030B nous informe sur les modes de fonctionnement des PORTS, des TIMERS et du registre à décalage. Il est en effet possible de verrouiller les données entrant sur les PORTS 1 et B par mise à 1 des bits correspondants de ACR.

Bit 0 consacré au PORT A, Bit 1 au PORT B.

Bit 7 et Bit 6 commandent le compteur T1.

Bit 5 commande le fonctionnement du compteur T2.

Les bits 2, 3, 4 commandent le registre à décalage.

Le registre à décalage est à l'adresse 030A. Ce registre peut être actionné par l'horloge interne ou par le compteur 2. 8 configurations sont ainsi autorisées.

LES TIMERS

Les timers incorporés au VIA permettent bien des utilisations, tant en entrée qu'en sortie. On pourra les utiliser pour générer des délais ou encore, mesurer des intervalles de temps.

Ils occupent 6 positions mémoire. 4 pour le Timer 1 et 2 pour le Timer 2. Ils fonctionnent sur 16 bits. Ils ne sont pas strictement identiques. leur mode de fonctionnement est défini par les bits 5, 6, 7 de ACR.

En mode monostable, on compte un nombre d'impulsions ou on mesure la durée d'une impulsion. Ce ci est valable pour T 1 et T2.

En mode oscillateur, on génère ou on compte un train d'impulsions continues. Ceci ne s'applique qu'à T1.

Le timer 2 peut compter des impulsions sur la broche PB6 (pas utilisable ainsi sur ORIC, PB6 étant câblé à la commande de relais).

Le timer 1 peut produire des impulsions sur la broche PB7 (utilisation ORIC : écriture cassette).

Ces utilisations des broches PB6 et PB7 sont prioritaires sur les emplois en PORT.

L'utilisation du timer (2 par exemple) en temporisateur, se fera en mode monostable.

La temporisation démarre quand on charge l'octet de poids fort du timer. Ceci remet également à 0 l'indicateur d'interruption correspondant. Le passage à zéro du compteur (qui fonctionne en fait en décompteur) provoque la mise à 1 du bit d'interruption de l'indicateur correspondant.

Il est important de noter que la lecture de l'octet de poids faible du provoque la remise à zéro de l'indicateur d'interruption.

Nous voyons là toute la puissance du matériel mis à notre disposition (ou tout au moins, en partie) par l'ORIC. Une application des TIMER est donnée dans le programme « FRÉQUENCE ».

En résumé, nous vous proposons deux tableaux regroupant les utilisations des diverses ressources du VIA de l'ORIC (tableau 1) et les emplacements mémoires et rôle des registres VIA.

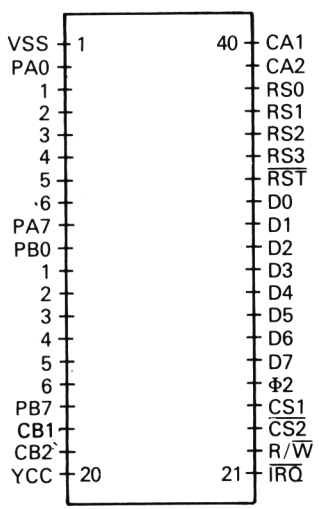
TABLEAU 1

PA0 ~ PA7	imprimante et données AY-8912 (générateur sonore)
CA1	ACKnoledge imprimante
CB1	lecture cassette
CA2	BC1 du AY-8912 } écriture, lecture et inhibition
CB2	BDIR du AY-8912 } du AY-8912
PB0 ~ PB2	clavier : matriçage lignes AD0 ~ AD2
PB3	strobe clavier
PB4	strobe imprimante
PB5	inutilisé
PB6	Commande relais télécommande magnéto
PB7	écriture sur cassette

TABLEAU 2

0300	ORB	sortie PORT B	Registres du PIO
0301	ORA	sortie PORT A	
0302	DDRB	direction PORT B	
0303	DDRA	direction PORT A	
0304	poids faible	écriture timer, écriture	Registres des TIMERS
0305	poids fort	lecture compteur (T1)	
0306	poids faible	accès timer latche	
0307	poids fort	(T1)	
0308	poids faible	lecture T2, RAZ interruption compteur écriture T2 sans RAZ	
0309	poids fort	écriture/lecture T2. RAZ interruption sur écriture	
030A	SR	Registre à décalage (E/S série)	
030B	ACR		
030C	PCR		
030D	IFR		
030E	IER		
030F	ORA sans handshake		

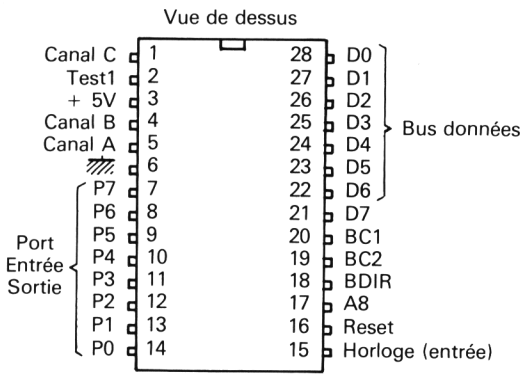
Toutes ces adresses sont situées, bien sûr, en page 3 (0300 à 03FF) de la mémoire, zone réservée aux entrées-sorties.



DÉSIGNATION DES BROCHES DU VIA. 6522

LE SON SUR L'ORIC

Dans votre ordinateur, un circuit spécialisé s'occupe de la génération des sons. C'est le AY3-8912 de General Instrument. Ce circuit complexe permet tout un tas de possibilités telles que musique synthétique, effets sonores, alarmes, génération de tonalités et modulateur FSK. Il possède également un port d'entrée-sortie permettant de dialoguer avec l'extérieur.



LE AY3-8912

ARCHITECTURE

L'ensemble du circuit est contrôlé par un ensemble de 15 registres.

BIT / REGISTRE	7	6	5	4	3	2	1	0	
0	Réglage fin tonalité canal A								8 bits utiles
1					Réglage gros				4 bits utiles
2	Réglage fin tonalité canal B								8 bits utiles
3					Réglage gros				4 bits utiles
4	Réglage fin tonalité canal C								8 bits utiles
5					Réglage gros				4 bits utiles
6				Période de bruit					5 bits utiles
7	AUTORISATIONS								8 bits utiles
8				Volume canal A					5 bits utiles
9				Volume canal B					5 bits utiles
10				Volume canal C					5 bits utiles
11	Réglage fin période enveloppe								8 bits utiles
12	Réglage gros période enveloppe								8 bits utiles
13					Sélection enveloppe				4 bits utiles
14	Données du Port								8 bits utiles

A QUOI SERVENT CES REGISTRES ?

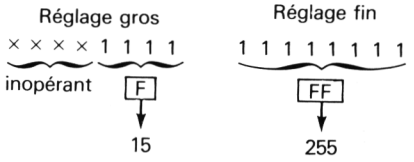
Nous allons essayer de les décrire l'un après l'autre.

REGISTRES DE TONALITÉ 0-1-2-3-4 et 5

Dans le AY3-8912, le signal d'horloge (1 MHz) venant de l'oscillateur à quartz est tout d'abord divisé par 16. On dispose donc d'une fréquence de 62,5 kHz. Il s'agit d'un signal carré qui peut être envoyé sur chaque canal directement ou bien divisé. Le facteur de division est donné pour chaque canal par le registre réglage fin et le registre réglage gros.

Exemple : On désire avoir sur le canal B, un signal de 2 500 Hz. Le facteur de division va donc être : $\frac{62,5}{2,5} = 25$ soit en Hexadécimal : 19_H. Il va donc falloir mettre dans le registre 2 la valeur 19_H et dans le registre 3 la valeur 0.

La valeur maximale du facteur de division est obtenue quand tous les bits des 2 registres (réglage fin et réglage gros) sont à 1. En fait, seuls les bits 0, 1, 2 et 3 du réglage gros sont utiles de sorte qu'au maximum on peut avoir



valeur max. : (15 × 256) + 255 = 4 095

D'où la fréquence minimale : $fmin = \frac{62,5}{4\,095} = 15,26$ Hertz

La fréquence maximale : fmax = 62,5 kHz.

REGISTRE DE BRUIT (6)

Tout comme pour les registres de tonalité, la fréquence d'horloge est d'abord divisée par 16. Seulement, le facteur de division programmable est limité dans ce cas à 5 bits soit 1F en hexadécimal et 31 en décimal de sorte que :

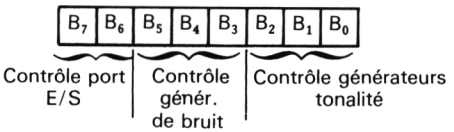
$$Fmax = \frac{F_H}{16} = \frac{1\ MHz}{16} = 62,5\ KHz$$

$$Fmin = \frac{62,5}{31} \simeq 2\ KHz$$

Il s'agit dans ce cas non pas d'un signal carré mais d'impulsions.

REGISTRE D'AUTORISATIONS (7)

C'est un registre important qui contrôle l'ensemble du circuit AY3-8912. Voici sa configuration :



Pour les bits 0 à 5, un zéro équivaut à une autorisation, un 1 à une inhibition.

B0 — bit contrôle générateur tonalité canal A
B1 — bit contrôle générateur tonalité canal B
B2 — bit contrôle générateur tonalité canal C
B3 — bit contrôle générateur bruit canal A
B4 — bit contrôle générateur bruit canal B
B5 — bit contrôle générateur bruit canal C

Le bit B6 sert à positionner le port d'entrée/sortie. Un 1 met tout le port en sortie, un 0 le met en entrée *(dans le cas de l'ORIC, ne jamais mettre B6 à 0 sinon perte s/lecture clavier).*

Le bit B7 ne sert à rien dans le AY 8912.

Exemple : Soit à générer sur les canaux A et B des tonalités et sur le port C un bruit. On aura donc à mettre dans le registre 7 :

B7	B6	B5	B4	B3	B2	B1	B0	
X	1	0	1	1	1	0	0	= 1B en hexadécimal

indifférent

Registres de volume (8-9-10)

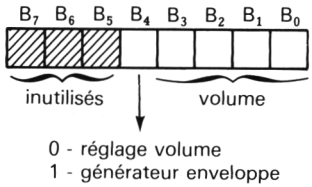
Ils permettent le réglage du volume sonore des canaux respectifs A-B et C.

Cinq bits sont utiles (B0 à B4).

Les bits B0 à B3 règlent le volume à proprement parler.

Le bit B4 sélectionne le générateur d'enveloppes quand il passe à 1.

Le volume se règle donc à l'aide de 4 bits. On a donc 16 possibilités de niveau sonore (de 0 à 15) et ce indépendamment pour chaque canal.

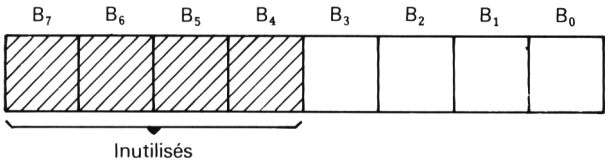


REGISTRE SÉLECTION D'ENVELOPPE (13)

Quand le bit B4 des registres de volume passe à 1, le niveau sonore n'est plus réglable manuellement mais dépend d'un autre générateur dit « d'enveloppes ». Celui-ci permet en somme d'avoir une modulation d'amplitude du signal BF issu des 3 canaux.

Les enveloppes possibles sont au nombre de 10 sélectionnées par les 4 bits B0-B1-B2 et B3 du registre 13.

FORMES D'ENVELOPPES



× = indifférent

Formes d'enveloppes

00	0	0	×	×	
04	0	1	×	×	
08	1	0	0	0	
09	1	0	0	1	
0A	1	0	1	0	
0B	1	0	1	1	
0C	1	1	0	0	
0D	1	1	0	1	
0E	1	1	1	0	
0F	1	1	1	1	

Période enveloppe

REGISTRES RÉGLAGE PÉRIODE ENVELOPPE (11-12)

La fréquence de l'enveloppe est obtenue à partir de la fréquence d'horloge (1 MHz) qui est au préalable divisée par 256. Le facteur de division est donné ensuite par les registres 11 (réglage fin) et 12 (réglage gros). On dispose donc de 16 bits, soit un facteur diviseur maximum de 65535.

Donc, pour la période la plage de réglage est :

$$Tmin = \frac{256}{62,5} = 4,1 \text{ ms}$$

$$Tmax = \frac{256 \times 65535}{62,5} = 268,5 \text{ secondes}$$

REGISTRE DE DONNÉES DU PORT (14)

Si le port est en sortie (sélection par le registre 7), les données à sortir sont à mettre dans le registre 14. Pour le port en entrée, les données sont à lire également dans le registre 14. Dans les 2 cas, il y a mémorisation des états.

Voilà donc la description du générateur sonore. Cependant, si son utilisation par le basic ne pose pas de problème grâce aux instructions SOUND, PLAY et MUSIC sur lesquelles nous ne reviendrons pas, il n'en va pas de même en langage machine. En effet, ce circuit est relié au micro-processeur par l'intermédiaire du VIA ce qui ne facilite pas son accès d'autant plus qu'il est également utilisé pour la scrutation clavier. Heureusement tout n'est pas perdu car il existe dans la ROM une routine permettant d'écrire dans les registres du AY 3-8912.

Cette routine se trouve à l'adresse hexadécimale F 535. Son utilisation est simple. Il suffit de charger avant l'appel de cette routine, le registre A avec le numéro du registre du circuit sonore dans lequel on veut écrire et dans le registre X la donnée. Après le retour du saut en # F 535, le registre sélectionné contiendra la donnée. **Attention cependant, deux mises en garde :**

* Dans cette routine, le registre Y est utilisé. Donc si vous l'utilisez dans votre programme principal, préserver-le au préalable.

* Si vous voulez conserver la « main » au clavier, n'utilisez pas le registre 14, il est utilisé par l'ORIC pour lire le clavier.

EXEMPLE D'UTILISATION

Générateur d'un signal continu à 6,25 kHz sur le canal 1 avec un volume moyen

$$(6,25 \text{ KHz} = \frac{62,5}{10} \Rightarrow \text{facteur division} = 10 = 0A_H)$$

```
LDA % # 01
LDX % # 00      Met 0 dans le registre 1 (réglage gros ton)
JSR # F 535
LDA % # 00
LDX % # 0A      Met 0A dans le registre 0 (réglage fin)
JSR # F 535
LDA % # 08
LDX % # 08      Met le volume en position moyenne (registre 8)
JSR # F 535
LDA % # 07
LDX % # FE      Autorise signal canal A (11111110)
JSR # F 535
RET
```

Pour essayer ce programme en langage machine, entrez le programme suivant :

```
10 FORN=0TO28
20 READA$:A=VAL("#"+A$):POKE#400+N,A
30 NEXT
50 DATAA9,01,A2,00,20,35,F5
60 DATAA9,00,A2,0A,20,35,F5
70 DATAA9,08,A2,08,20,35,F5
80 DATAA9,07,A2,FE,20,35,F5
90 DATA60
```

Faites RUN.

Ensuite, quand vous ferez CALL # 400, la tonalité de 6,25 kHz sera générée (pour arrêter : CTRL F).

LE MICROPROCESSEUR ET L'ASSEMBLEUR

- LE 6502
- INTRODUCTION A L'ASSEMBLEUR
- L'ASSEMBLEUR DU 6502

LE MICROPROCESSEUR 6502

Fabriqué par Rockwell le 6502 est un microprocesseur de type 8 bits. Nous allons examiner très succinctement ses caractéristiques en renvoyant le lecteur, désireux de dépasser ce stade élémentaire, vers un ouvrage plus complet.

Vu par le programmeur, le microprocesseur est essentiellement un ensemble de registres qui sont :

1 - L'accumulateur (A) point de transit de prédilection pour les données arithmétiques et logiques. Beaucoup d'instructions lui sont consacrées, permettant par un adressage « implicite » une grande rapidité d'exécution.

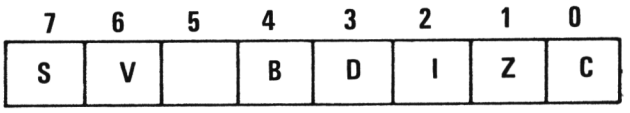
2 - Deux registres d'index (X et Y) ayant un rôle bien particulier dans l'adressage indexé (voir description des différents modes d'adressage), mais qui peuvent aussi être utilisés pour diverses tâches par le programmeur. Comme l'accumulateur ces 2 registres sont sur 8 bits.

3 - Le compteur programme (compteur ordinal)(PC) est incrémenté de 1 à chaque octet trouvé par le microprocesseur ce qui permet à ce dernier de connaître à tout instant l'adresse de la prochaine instruction. Ce registre est un registre 16 bits puisqu'il manipule des adresses.

4 - Le pointeur de pile (SP) est également un registre au rôle essentiel. Nous savons que l'exécution séquentielle d'un programme peut être déroutée vers un sous-programme. Pour s'y retrouver, on mémorise l'adresse de retour dans une zone mémoire particulier, appelée PILE. Cette zone sur ORIC est entre 0100 et 01FF. Les adresses étant sur 16 bits, elles occupent 2 octets. On pourra donc mémoriser jusqu'à 128 adresses. Le pointeur de pile tient à jour, pour renseigner le microprocesseur, la première adresse de la pile.

5 - Le dernier registre qui retiendra notre attention est **le registre d'état.** Sur les 8 bits disponibles, 7 sont utilisés. Ce sont des indicateurs (« flags » en anglais, « drapeaux » en français). Ils prendront la valeur 0 ou 1 suivant certaines conditions. Ils seront très utilisés par le programme, surtout pour les tests de branchements.

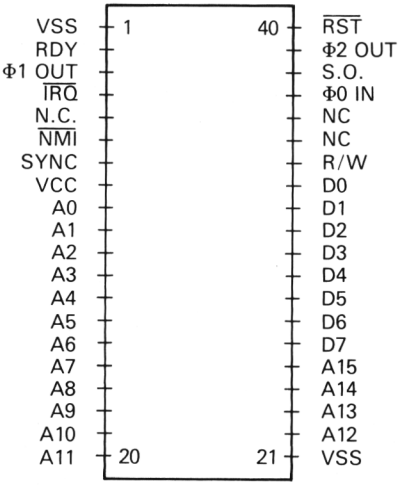
La configuration du registre est



C est le bit de retenue (correspondant à la retenue issue du bit 7)
Z est l'indicateur de zéro (vaut 1 si le résultat d'une opération est nul)
I inhibe les interruptions s'il est à 1
D indicateur de mode décimal (si à 1)
B indique qu'une PAUSE (BRK) a été exécutée (si 1)
V indicateur de débordement
S indicateur de signe

Le bit 5 n'est pas utilisé.

Le microprocesseur, outre ses lignes de contrôle, possède un bus de données, sur 8 bits et un bus d'adresses sur 15 bits. Il sera capable d'adresser 64 octets de mémoire.



Désignation des broches du microprocesseur 6502

INTRODUCTION A L'ASSEMBLEUR

La programmation en langage BASIC ayant ses limites que nous avons vues antérieurement, il est souvent utile de programmer en langage machine soit directement en hexadécimal, soit par l'intermédiaire d'un programme d'assemblage.

Quel que soit votre choix, vous aurez recours à l'assembleur, au moins lors de l'écriture du programme sur papier. Seule différence, le programme d'assemblage se chargera de la transformation en code objet (langage machine) et effectuera les calculs de branchements. Ce travail, vous devrez l'effectuer si vous ne disposez pas d'un tel programme.

Nous avons voulu permettre au lecteur de faire ses premiers pas dans ce domaine sans devoir, pour autant acquérir d'autres ouvrages, c'est pourquoi vous trouverez dans ce livre toute la liste des instructions 6502, ainsi que leurs effets. Vous serez donc à même d'écrire vos premiers programmes.

PRÉSENTATION D'UN LISTING EN ASSEMBLEUR

Un listing en assembleur est séparé en plusieurs zones, appelées champs. Apprendre à respecter ces zones, même lors de l'écriture d'un programme sur papier, est une bonne habitude.

Nous trouverons ainsi les champs :

Adresse
Objet
Instructions
Commentaires

Le champ « adresse », comme son nom l'indique contient les adresses où sont implantées le instructions. Ces adresses seront, le plus souvent, exprimées en hexadécimal.

Le champ « objet » contient les octets de langage machine correspondant aux diverses instructions.

Le champ « instructions » contiendra les instructions sous forme de mnémoniques. Le mnémonique permet de retrouver aisément le rôle d'une instruction à partir de l'abréviation de son nom anglais. Ainsi :

Load (charger) sera LD et on aura
LDA pour chargement de l'accumulateur
Store (mémoriser) sera ST et on aura
STA pour mémorisation du contenu de l'accumulateur
Junp (sauter) sera JMP, etc...

Le champ « commentaires » contiendra toutes les informations nécessaires à la bonne compréhension du programme par l'utilisation ou par un autre programmeur.

Vous aurez tout intérêt à ne pas négliger ce travail de renseignement de votre listing assembleur : ceci vous permettra un gain de temps appréciable lors de modifications futures du programme.

Vous trouverez aussi un champs « étiquette » situé entre les champs « adresses » et « objet » ou devant le champs « instructions » permettant de repérer plus aisément les branchements ou les sous-programmes.

L'aspect d'un listing assembleur pourra être le suivant :

Adresses	Objet	Etiquettes	Mnémoniques	Commentaires
8000	AD 00 03	LECT	LDA 0300	Lit le port B VIA
8003	29 10		AND % 10	Teste bit 4
8005	F0 01		BEQ PREPAR	Saut si bit 4 à 1
8007	60		RTS	Retour sinon
8008	A9 FF	PREPAR	LDA % # 42	Met le code ASCII de lettre « B » dans accu
800A	AA		TAX	Transféré dans X
800B	60		RTS	Revient

La programmation en assembleur implique une bonne connaissance des différents modes d'adressage du microprocesseur ainsi que de l'effet produit, notamment sur les indicateurs (« drapeaux » ou « flags »), par les diverses instructions utilisées.

Une programmation efficace sera obtenue par la recherche du moindre nombre d'instructions, ainsi que par l'emploi des instructions les plus rapides.

Dans le cas de l'ORIC, ceci n'est pas toujours possible car les possibilités d'adressage en page zéro sont restreintes par la présence de variables contrôlant le fonctionnement du système dans cette zone.

Tous les renseignements dont vous aurez besoin sont consignés dans les tableaux résumant les diverses instructions du 6502.

L'ASSEMBLEUR DU 6502

Le 6502 a 56 instructions dont certaines se divisent en sous-groupes suivant le mode d'adressage.

LES MODES D'ADRESSAGE SONT :

- Implicite
- Accumulateur
- Immédiat
- Page zéro
- Page zéro indexé
- Absolu
- Absolu indexé
- Indirect indexé

L'adressage implicite : se dit pour les instructions qui se suffisent à elle-mêmes. *Ex :* CLC - DEY.

Elles sont codées sur un octet.

L'adressage accumulateur : ne concerne que les instructions relatives à l'accumulateur. *Ex :* ROL A - ASL A.

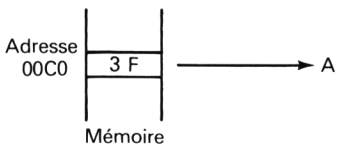
L'adressage immédiat : signifie que la donnée nécessaire à l'instruction se trouve immédiatement après. *Ex :* LDA # % FD

instruction donnée

Le « % » signifie que la donnée suit.

L'adressage page zéro : cette fois-ci, le code qui suit l'instruction n'est pas une donnée mais une adresse sur 1 octet (0 à 255) donnant l'adresse en mémoire où se trouve la donnée et ce, en page 0 car l'adresse ne peut valoir au maximum que FF_H (255).

Ex : LDA # C0

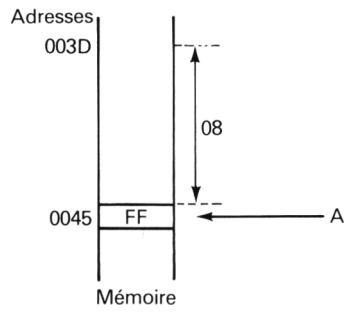


Après l'instruction : A = 3 F

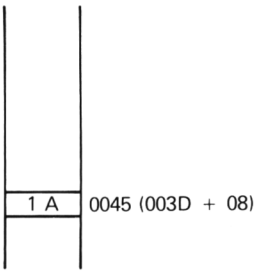
L'adressage page zéro indexé : même procédure que précédemment sauf que l'adresse où aller chercher la donnée est celle qui suit l'instruction à laquelle on additionne un index. Cet index peut être suivant le cas le registre X ou Y.

Ex : STA # 3D, X

Avant l'instruction : A = 1A X = 08



Après l'instruction : A = 1A X = 08



L'adressage absolu : de même que pour l'adressage page 0, il est nécessaire de spécifier l'adresse où se trouve la donnée mais en adressage absolu, toute la zone mémoire est disponible de sorte que 2 octets sont utiles.

Ex : CPX # FBE0

En FBE0 se trouve la donnée à traiter.

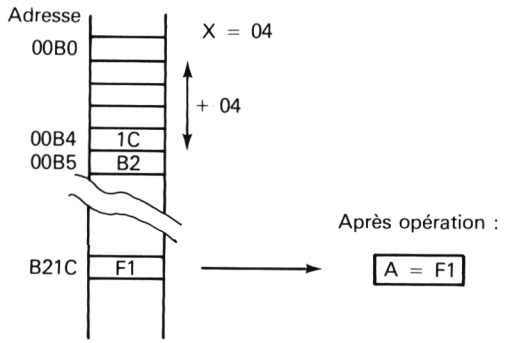
L'adressage absolu indexé : on rajoute à l'adresse donnée, un index qui peut être le registre X ou Y suivant le cas.

Ex : LDX # 10A3, Y

A l'adresse 10A3 + Y se trouve la donnée à mettre dans X.

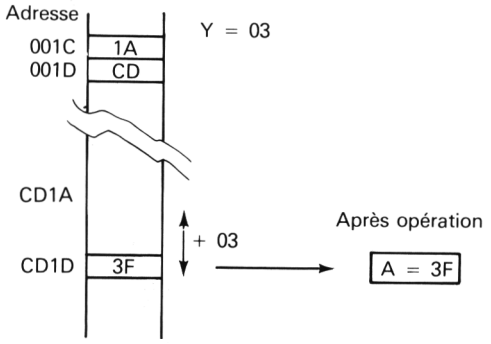
L'adressage indirect indexé X : ne fonctionne qu'en page zéro. Cet adressage s'appelle aussi pré-indexé. Il consiste à aller chercher à l'adresse donnée (sur 1 octet car elle est en page 0) augmentée de la valeur du registre d'index X, une autre adresse, qui elle, est sur 2 octets, où se trouve la donnée.

Ex : LDA # B0, X) notez les parenthèses signigiant que l'indexation a lieu sur B0.



L'adressage (indirect) indexé Y : ou post-indexé. Ici, on va chercher à l'adresse donnée (toujours sur 1 octet car en page zéro), une autre adresse sur 2 octets à laquelle on rajoute la valeur du registre index Y, afin de récupérer la donnée nécessaire à l'instruction. Ceci explique pourquoi le terme « POST-INDEXÉ ».

Ex : LDA (# 1C), Y.



UTILISATION DES TABLEAUX D'INSTRUCTIONS

Chaque instruction du 6502 est décrite sous ses différentes formes d'adressage, dans un tableau résumant ses caractéristiques essentielles.

Vous trouverez ainsi :
— l'opération réalisée,
— sa description, sous forme résumée,
— l'action de l'instruction sur les indicateurs du registre d'état.

Puis, pour les différents modes d'adressage :

— le nombre de cycles machine, permettant de déterminer le temps demandé pour la complète exécution de l'instruction. Dans le cas de l'ORIC-1, ce temps est obtenu en multipliant le nombre de cycles requis par la période de l'horloge qui est de 1 μs,
— le nombre d'octets associés à l'instruction,
— le code hexadécimal correspondant.

Note : en ce qui concerne le nombre de cycles machines vous trouverez les notations 5/6 qui signifie que l'instruction peut prendre 5 cycles si elle est contenue dans la même page mémoire ou 6, si on change de page.

2+ signifie 3 périodes d'horloge pour un branchement dans la même page, 4 pour une autre page.

ADC (Addition avec retenue)

Opération : A ← A + M + C

Description : réalise l'addition entre l'accumulateur, l'opérande et la retenue. Le résultat se retrouve dans l'accumulateur.

Drapeaux positionnés :

N = 1 si le bit 7 de l'accumulateur après opération est à 1
V = 1 si débordement
Z = 1 si tous les bits du résultat sont à 0
C = 1 si il y a une retenue

Mode d'adressage	Nombre de cycles machine	Nombre d'octet	Code hexa.
Immédiat	2	2	69
Page zéro	3	2	65
Page zéro indexé X	4	2	75
Absolu	4	3	6D
Absolu indexé X	4/5	3	7D
Absolu indexé Y	4/5	3	79
Indirect indexé X	6	2	61
(Indirect) indexé Y	5/6	2	71

AND (« ET » logique)

Opération : A ← A.M

Description : réalise un « ET » logique entre le contenu de l'accumulateur et le contenu de la mémoire. Le résultat se retrouve dans A (chaque bit de l'accumulateur après opération sera le résultat du « ET » logique entre les bits correspondants de A et M avant opération).

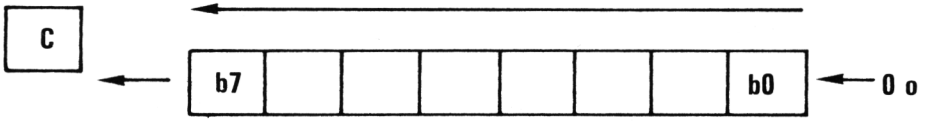
DRAPEAUX POSITIONNÉS

N = 1 si le bit 7 de A est à 1 après opération
Z = 1 si le résultat est nul

Mode d'adressage	Nombre de cycles machine	Nombre d'octet	Code hexa.
Immédiat	2	2	29
Page zéro	3	2	25
Page zéro indexé X	4	2	35
Absolu	4	3	2D
Absolu indexé X	4/5	3	3D
Absolu indexé Y	4/5	3	39
Indirect indexé X	6	2	31
(Indirect) indexé Y	5/6	2	31

ASL (Décalage arithmétique à gauche)

Opération :



Description : Décalage de tous les bits de l'accumulateur ou de l'opérande d'un cran à gauche. Le bit 7 va dans la retenue, le bit 0 est mis à 0.

DRAPEAUX POSITIONNÉS

N = 1 si le bit 7 est à 1 après opération
Z = 1 si le résultat est 0
C = 1 si la retenue est à 1 après opération

Mode d'adressage	Nombre de cycles machine	Nombre d'octet	Code hexa.
Accumulateur	2	1	0A
Page zéro	5	2	06
Page zéro indexé X	6	2	16
Absolu	6	3	0E
Absolu indexé X	7	3	1E

BCC (Saut si non-retenue)

Opération : PC ← PC + 0002 + Saut si C = 0

Description : Teste si la retenue est à zéro. Si oui, saut à l'adresse présente augmentée de 2 et de la valeur du saut (+ ou −).

$0 \leqslant$ Saut $\leqslant 127 \rightarrow$ Saut positif
$128 <$ Saut $\leqslant 255 \rightarrow$ Saut négatif

Drapeaux positionnés :

Aucun

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Relatif	2+	2	90

BCS (Saut si retenue)

Opération : PC ← PC + 0002 + Saut si C = 1

Description : Saut si la retenue est à 1

Drapeaux positionnés :

Aucun

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Relatif	2+	2	B0

BEQ (Saut si égal)

Opération : PC ← PC + 0002 + Saut si Z = 1

Description : Saut si Z = 1

Drapeaux positionnés :

Aucun

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Relatif	2+	2	F0

BIT (Test bit)

Opération : A.M

Description : « ET » logique respectivement entre tous les bits de l'accumulateur et l'opérande. Le résultat n'est pas conservé mais positionne les drapeaux.

Drapeaux positionnés :

N = 1 si le bit 7 après opération est à 1
V = 1 si le bit 6 après opération est à 1
Z = 1 si tous les bits après opération sont à 0

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Page zéro	3	2	24
Absolu	4	3	2C

BMI (Saut si négatif)

Opération : PC ← PC + 0002 + Saut si Z = 1

Description : Saut si le résultat est négatif (bit 7 = 1)

Drapeaux positionnés :

Aucun

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Relatif	2+	2	30

BNE (Saut si non-égal)

Opération : PC ← PC + 0002 + Saut si Z = 0

Description : Saut si Z = 0

Drapeaux positionnés :

Aucun

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Relatif	2+	2	D0

BPL (Saut si positif)

Opération : PC ← PC + 0002 + Saut si N = 0

Description : Saut si N = 0 (bit 7 = 0)

Drapeaux positionnés :

Aucun

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Relatif	2+	2	10

BRK (Arrêt)

Opération : SP ← PC + 0002 SP + 1 ← P

Description : Le programme est arrêté, le registre d'état et le compteur programme sont sauvegardés.

Drapeaux positionnés :

I = 1

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	7	1	00

BVC (Saut si non-débordement)

Opération : PC ← PC + 0002 + Saut si V = 0

Description : Saut si V = 0

Drapeaux positionnés :

Aucun

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Relatif	2+	2	50

BVS (Saut si débordement)

Opération : PC ← PC + 0002 + Saut si V = 1

Description : Saut si V = 1

Drapeaux positionnés :

Aucun

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Relatif	2+	2	70

CLC (Effacement retenue)

Opération : C ← 0

Description : Effacement de la retenue dans le registre d'état

Drapeaux positionnés :

C = 0

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	2	1	18

CLD (Annulation du mode décimal)

Opération : D ← 0

Description : Mise à zéro du mode décimal
Remet le microprocesseur en mode binaire

Drapeaux positionnés :

D = 0

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	2	1	D8

CLI (Autorisation des interruptions)

Opération : I ← 0

Description : Les interruptions sont autorisées

Drapeaux positionnés :

I = 0

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	2	1	58

CLV (Effacement du bit de débordement)

Opération : V ← 0

Description : Remet à zéro le bit de débordement

Drapeaux positionnés :

V = 0

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	2	1	B8

CMP (Comparaison)

Opération : A-M

Description : Comparaison entre l'accumulateur et l'opérande. Il s'agit en fait d'une soustraction (A-M) dont le résultat n'est pas conservé mais qui sert à positionner les drapeaux.

Drapeaux positionnés :

N = 1 si le bit 7 est à 1 après l'opération
Z = 1 si tous les bits sont à 0 (Identicité) après opération
V = 1 si débordement
C = 1 si l'opérande est supérieur à l'accumulateur

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Immédiat	2	2	C9
Page zéro	3	2	C5
Page zéro indexé X	4	2	D5
Absolu	4	3	CD
Absolu indexé X	4/5	3	DD
Absolu indexé Y	4/5	3	D9
Indirect indexé X	6	2	C1
(Indirect) indexé Y	5/6	2	D1

CPX (Comparaison registre index)

Opération : X-M

Description : Idem que CMP avec le registre X

Drapeaux positionnés :

N = 1 si le bit 7 est à 1 après opération
Z = 1 si équivalent à l'opérande
C = 1 si l'opérande est supérieur à X

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Immédiat	2	2	E0
Page zéro	3	2	E4
Absolu	4	3	EC

CPY (Comparaison registre index)

Opération : Y-M

Description : Idem que CMP avec le registre X

Drapeaux positionnés :

N = 1 si le bit 7 est à 1 après l'opération
Z = 1 si équivalence
C = 1 si l'opérande est supérieur à Y

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Immédiat	2	2	C0
Page zéro	3	2	C4
Absolu	4	3	CC

DEC (Décrémentation)

Opération : M ← M-1

Description : Soustraction de 1 à la valeur de l'opérande

Drapeaux positionnés :

N = 1 si le bit 7 est à 1 après opération
Z = 1 si le résultat est zéro

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Page zéro	5	2	C6
Page zéro indexé X	6	2	D6
Absolu	6	3	CE
Absolu indexé X	7	3	DE

DEX (Décrémentation registre indexé)

Opération : X ← X-1

Description : Idem DEC avec registre X

Drapeaux positionnés :

N = 1 si le bit 7 est à 1
Z = 1 si le résultat est zéro

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	2	1	CA

DEY (Décrémentation registre indexé)

Opération : Y ← Y-1

Description : Idem DEC avec registre Y

Drapeaux positionnés :

N = 1 si le bit 7 est à 1 après opération
Z = 1 si le résultat est zéro

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	2	1	88

EOR (OU EXCLUSIF)

Opération : A ← A ⊕ M

Description : Réalise un « OU Exclusif » entre le contenu de l'accumulateur et le contenu de l'opérande bit à bit
Rappel : 1 ⊕ 1 = 0

Drapeaux positionnés :

N = 1 si bit 7 = 1
Z = 1 si le résultat est nul

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Immédiat	2	2	49
Page zéro	3	2	45
Page zéro indexé X	4	2	55
Absolu	4	3	4D
Absolu indexé X	4/5	3	5D
Absolu indexé Y	4/5	3	59
Indirect indexé X	6	2	41
(Indirect) indexé Y	5/6	2	51

INC (Incrémentation)

Opération : M ← M + 1

Description : Addition de 1 à la valeur de l'opérande

Drapeaux positionnés :

N = 1 si le bit 7 = 1
Z = 1 si le résultat est nul

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Page zéro	5	2	E6
Page zéro indexé X	6	2	F6
Absolu	6	3	EE
Absolu indexé X	7	3	FE

INX (Incrémentation registre index)

Opération : X ← X + 1

Description : Idem que INC avec le registre X

Drapeaux positionnés :

N = 1 si le bit 7 = 1
Z = 1 si résultat est nul

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	2	1	E8

INY (Incrémentation registre index)

Opération : Y ← Y + 1

Description : Idem que INC avec le registre Y

Drapeaux positionnés :

N = 1 si le bit 7 = 1
Z = 1 si résultat est nul

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	2	1	C8

JMP (Saut non relatif)

Opération : PC ← adresse

Description : Saut inconditionnel à l'adresse spécifiée

Drapeaux positionnés :

Aucun

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Absolu	3	3	4C
Indirect	5	3	6C

JSR (Saut à un sous-programme)

Opération : SP ← PC + 2 – PC ← adresse

Description : Le compteur programme est incrémenté de deux puis sauvegardé dans la pile. Le saut est ensuite fait à l'adresse donnée

Drapeaux positionnés :

Aucun

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Absolu	6	3	20

LDA (Chargement accumulateur)

Opération : (SP) ← A

Description : Chargement de l'accumulateur

Drapeaux positionnés :

N = 1 si bit 7 est à 1 après opération
Z = 1 si tous les bits sont à zéro

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Immédiat	2	2	A9
Page zéro	3	2	A5
Page zéro indexé X	4	2	B5
Absolu	4	3	AD
Absolu indexé X	4/5	3	BD
Absolu indexé Y	4/5	3	B9
Indirect indexé X	6	2	A1
(Indirect) indexé Y	5/6	2	B1

LDX (Chargement registre X)

Opération : X ← M

Description : Chargement du registre X

Drapeaux positionnés :

N = 1 si bit 7 = 1
Z = 1 si X = 0

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Immédiat	2	2	A2
Page zéro	3	2	A6
Page zéro indexé Y	4	2	B6
Absolu	4	3	AE
Absolu indexé Y	4/5	3	BE

LDY (Chargement registre Y)

Opération : Y ← M

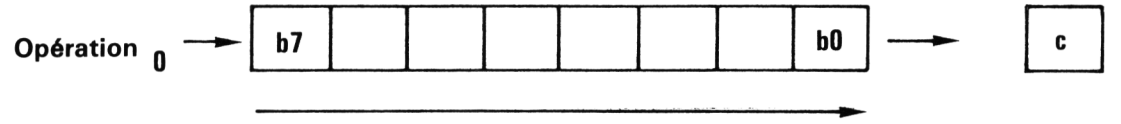
Description : Chargement registre Y

Drapeaux positionnés :

Z = 1 si Y = 0

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Immédiat	2	2	A0
Page zéro	3	2	A4
Page zéro indexé X	4	2	B4
Absolu	4	3	AC
Absolu indexé X	4	3	BC

LSR (Décalage logique à droite)



Description : Décalage de tous les bits de l'opérande d'un cran à droite. La retenue reçoit le bit 0 alors que le bit 7 reçoit un zéro

Drapeaux positionnés :

N = 0
Z = 1 si le résultat est nul
C = 1 si b0, avant l'opération, était à 1

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Accumulateur	2	1	4A
Page zéro	5	2	46
Page zéro indexé X	6	2	56
Absolu	6	3	4E
Absolu indexé X	7	3	5E

NOP (Non-Opération)

Opération : aucune

Description : Pas d'opération

Drapeaux positionnés :

Aucun

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	2	1	EA

ORA (OU logique)

Opération : A ← A.M

Description : Réalise un « OU » logique entre chaque bit de l'accumulateur et de l'opérande. Le résultat se retrouve dans A

Drapeaux positionnés :

N = 1 si bit 7 = 1
Z = 1 si résultat nul

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Immédiat	2	1	09
Page zéro	3	2	05
Page zéro indexé X	4	2	15
Absolu	4	3	0D
Absolu indexé X	4/5	3	1D
Absolu indexé Y	4/5	3	19
Indirect indexé X	6	2	01
(Indirect) indexé Y	5/6	2	11

PHA (Préserve l'accumulateur)

Opération : (SP) ← M

Description : Sauvegarde de l'accumulateur dans la pile.

Drapeaux positionnés :

Aucun

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	3	1	48

PHP (Préserve le registre d'états)

Opération : (SP) ← P

Description : Sauvegarde du registre d'états dans la pile.

Drapeaux positionnés :

Aucun

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	3	1	08

PLA (Récupère l'accumulateur)

Opération : A ← (SP)

Description : Transfert du contenu de la pile, adressé par le Stack pointeur, dans l'accumulateur

Drapeaux positionnés :

N = 1 si bit 7 = 1
Z = 1 si A = 0 après opération

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	4	1	68

PLP (Récupère le registre d'état)

Opération : P ← (SP)

Description : Transfert du contenu de la pile, adressé par SP dans le registre d'état

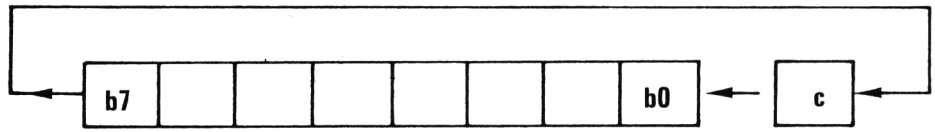
Drapeaux positionnés :

Tous rétablis

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	4	1	28

ROL (Rotation à gauche)

Opération :



Description : Rotation de tous les bits de l'accumulateur ou de l'opérande d'un cran à gauche avec la retenue allant dans le bit 0 et le bit 7 dans la retenue

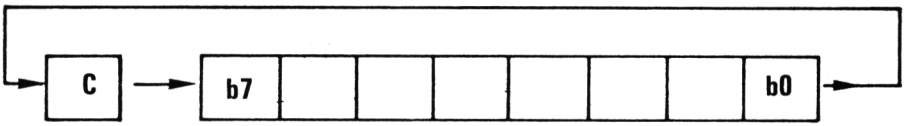
Drapeaux positionnés :

N = 1 si le bit 7 = 1 après opération
Z = 1 si le résultat est nul
C = 1 si la retenue est à 1

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Accumulateur	2	1	2A
Page zéro	5	2	26
Page zéro indexé X	6	2	36
Absolu	6	3	2E
Absolu indexé X	7	3	3E

ROR (Rotation à droite)

Opération :



Description : Rotation de tous les bits de l'accumulateur ou de l'opérande d'un cran à droite avec la retenue allant dans le bit 7 et le bit 0 dans la retenue

Drapeaux positionnés :

N = 1 si le b7 = 1
Z = 1 si résultat est nul
C = 1 si la retenue est à 1

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Accumulateur	2	1	6A
Page zéro	5	2	66
Page zéro indexé X	6	2	76
Absolu	6	3	6E
Absolu indexé X	7	3	7E

RTI (Retour d'interruption)

Opération : PC ← (SP)

Description : Le programme redémarre là où il avait été interrompu

Drapeaux positionnés :

Rétablis tels qu'ils étaient avant interruption

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	6	1	40

RTS (Retour de sous-programme)

Opération : PC ← (SP)

Description : Le programme principal repart juste après l'appel au sous-programme.

Drapeaux positionnés :

Aucun

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	6	1	60

SBC (Soustraction avec retenue)

Opération : A ← A-M-C

Description : Soustraction au contenu de A, du contenu de l'opérande et de la retenue. Le résultat se retrouve dans A.

Drapeaux positionnés :

N = 1 si bit 7 = 1
V = 1 si débordement
Z = 1 si résultat nul
C = 1 si la retenue est à 1

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Immédiat	2	2	E9
Page zéro	3	2	E5
page zéro indexé X	4	2	F5
Absolu	4	3	ED
Absolu indexé X	4/5	3	FD
Absolu indexé Y	4/5	3	F9
Indirect indexé X	6	2	E1
(Indirect) indexé Y	5/6	2	F1

SEC (Mise à 1 de la retenue)

Opération : C ← 1

Description : Mise à 1 de la retenue

Drapeaux positionnés :

C = 1

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	2	1	38

SED (Mise en mode décimal)

Opération : D ← 1

Description : Mise en mode décimal

Drapeaux positionnés :

D = 1

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	2	1	F8

SEI (Interdiction des interruptions)

Opération : I ← 1

Description : Interdiction des interruptions

Drapeaux positionnés :

I = 1

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	2	1	78

STA (Stockage de l'accumulateur)

Opération : M ← A

Description : Stocké le contenu de l'accumulateur en mémoire. A est inchangé.

Drapeaux positionnés :

Aucun

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Page zéro	3	2	85
Page zéro indexé X	4	2	95
Absolu	4	3	8D
Absolu indexé X	5	3	9D
Absolu indexé Y	5	3	99
Indirect indexé X	6	2	81
(Indirect) indexé Y	6	2	91

STX (Stockage du registre X)

Opération : M ← X

Description : Stocké le contenu du registre X en mémoire. X est inchangé.

Drapeaux positionnés :

Aucun

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Absolu	4	3	8E
Page zéro	3	2	86
Page zéro indexé Y	4	2	96

STY (Stockage du registre Y)

Opération : M ← Y

Description : Stocké le contenu du registre Y en mémoire. Y est inchangé.

Drapeaux positionnés :

Aucun

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Absolu	4	3	8C
Page zéro	3	2	84
Page zéro indexé X	4	2	94

TAX (Transfert de l'accumulateur dans le registre X)

Opération : X ← A

Description : Le contenu de l'accumulateur est transféré dans le registre X. A est inchangé.

Drapeaux positionnés :

N = 1 si b7 = 1
Z = 1 si A = 0

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	2	1	AA

TAY (Transfert de l'accumulateur dans le registre Y)

Opération : Y ← A

Description : Le contenu de A est transféré dans le registre Y. A est inchangé.

Drapeaux positionnés :

N = 1 si b7 = 1
Z = 1 si A = 0

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	2	1	A8

TSX (Transfert du « stack pointeur » plus 1 dans le registre X)

Opération : X ← S + 1

Description : Le contenu du « stack pointeur » est tout d'abord incrémenté puis transférer dans X tout en restant inchangé

Drapeaux positionnés :

N = 1 si b7 = 1
Z = 1 si S = 0

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	2	1	BA

TXA (Transfert du registre X dans A)

Opération : A ← X

Description : Le contenu de X est transféré dans A. X reste inchangé

Drapeaux positionnés :

N = 1 si b7 = 1
Z = 1 si X = 0

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	2	1	8A

TXS (Transfert du registre X dans le « Stack pointeur »

Opération : S ← X

Description : Le contenu du registre X est transféré dans le « stack pointeur ». X reste inchangé

Drapeaux positionnés :

Aucun

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	2	1	9A

TYA (Transfert du registre Y dans l'accumulateur)

Opération : A ← Y

Description : Le contenu du registre Y est transféré dans l'accumulateur. Y reste inchangé.

Drapeaux positionnés :

N = 1 si b_7 = 1
Z = 1 si Y = 0

Mode d'adressage	Nombre de cycles machine	Nombre d'octets	Code hexa.
Implicite	2	1	98

LES PROGRAMMES

— UTILITAIRES :

DÉSASSEMBLEUR
CALCUL D'ATTÉNUATEURS
MESURE DE FRÉQUENCE

— TRAFIC :

MIRE
LOCATORIC
CONTEST

— COMMUNICATION :

MONIMORSE	(moniteur de morse)
MANIPORIC	(manipulateur électronique)
ORICMORSE	(émission et réception MORSE)
RTTY	(émission et réception RTTY)
MAGNÉTOPHONE	

PROGRAMME DE DUMP CARACTÈRE ET DÉSASSEMBLAGE

Pour plusieurs raisons il peut être nécessaire de lister une zone mémoire de la machine (en ROM ou en RAM) ou de désassembler les routines qui s'y trouvent implantées. On peut en effet, vouloir rechercher en mémoire des chaînes de caractères pour, par exemple les modifier (textes commentaires en anglais à passer en français) ou s'inspirer de routines existantes pour en créer d'identiques.

La première fonction est appelée DUMP, la seconde est le DÉSASSEMBLAGE.

Ces deux tâches ont été confiées à un même programme, écrit en BASIC pour que sa structure soit bien apparente au lecteur. Malgré ce handicap la vitesse de travail est très acceptable, ce qui a été obtenu il est vrai, au prix de quelques concessions.

La principale est malheureusement liée à une « tare » du BASIC ORIC qui fait que, lors de la conversion d'un nombre en hexadécimal et de l'impression du résultat, le zéro de tête n'est pas affiché. Ce n'est pas gênant puisqu'il n'est pas significatif mais cela nuit un peu à la présentation d'un listing désassemblé. Ainsi on aura :

PRINT HEX $ (10) → # A au lieu de # 0A

Pour zéro, c'est plus ennuyeux car on ne retrouve que le seul signe # avec des blancs derrière.

Nous avons accepté ce handicap pour ne pas devoir écrire un sous-programme de conversion Décimal → Hexadécimal, les appels à ce sous-programme auraient considérablement ralenti l'exécution du programme principal.

Il faudra donc se souvenir que, un signe # tout seul représente un nombre nul, l'écriture # C représentant la forme hexadécimale du nombre 12.

Un autre problème, rencontré lors de l'élaboration de ce programme est lié aux difficultés de tabulation sur le BASIC ORIC.

Nous avons donc utilisé la variable système localisée en # 269 (n° de colonne de la prochaine impression) pour pallier ce défaut, en forçant le numéro de colonne par un POKE.

CONCEPTION D'UN DÉSASSEMBLEUR

Réaliser un programme de désassemblage est une tâche relativement aisée quand on sait comment sont organisées les suites d'octets que l'on trouve dans la mémoire de la machine.

Les instructions du 6502 peuvent utiliser jusqu'à 3 octets mais seul le premier est significatif de l'instruction, les autres caractérisant l'opérande.

Prenons des exemples :

— A9 1C signifie LDA % # 1C

Rappelons que le signe % est utilisé pour annoncer une donnée immédiate.
(Chargement de l'accumulateur avec la donnée 1C)

— 8D 10 90 signifie STA # 9010

Rappelons que les nombres sur deux octets sont représentés octet le moins significatif en tête.
(Rangement du contenu de l'accumulateur à l'adresse 9010)

— 6A signifie ROR

Rotation, par la droite, de l'accumulateur.

Le principe du programme est extrêmement simple puisqu'il suffit d'effectuer la reconnaissance sur le premier octet, le décodage de la suite s'effectuant en fonction de ce premier octet.

Les codes instruction seront donc rangés dans une table, en fonction de leur rang. En regard, figurera l'instruction. Tous les emplacements de la table ne seront pas utilisés.

PRINCIPE DU DÉCODAGE

Le programme désassembleur est donc lancé à l'adresse de début. Le premier octet lu va l'aiguiller sur un emplacement de la table qui contiendra l'instruction sous forme de chaîne de caractères.

Là, une petite astuce est utilisée. Le premier caractère de la chaîne n'appartient pas au mnémonique : c'est un chiffre de 0 à 9 qui indique à quelle « Famille » appartient l'instruction. En effet, nous avons classé les instructions selon leur mode d'adressage plus quelques exceptions.

Le travail est donc réalisé. L'instruction est complètement décodée par un sous-programme spécifique de sa famille.

L'impression à l'écran se fera suivant le format standard d'un listing en assembleur. A gauche l'adresse de l'instruction, au centre le code objet (1 à 3 octets), à droite le mnémonique correspondant.

Les octets ne correspondant pas à une instruction sont remplacés par le caractère ASCII équivalent ou par un point si c'est un caractère de contrôle.

L'appui maintenu sur la barre ESPACE interrompt le listage.

PRINCIPE DU DUMP CARACTÈRES

Il est beaucoup plus simple puisqu'il ne s'agit que de lister une zone mémoire sous forme de caractères ASCII équivalents. Seuls les codes n'ayant pas de correspondance sous forme de caractère, sont imprimés comme un point.

LE LISTING EN DÉTAIL

100-120 : création et remplissage de la table avec les mnémoniques rangés en DATA.
130 : passage en 40 colonnes.
150-160 : choix de la fonction.
190 : N et N2 sont les bornes du désassemblage.
200-202 : lecture et impression de l'adresse.
205 : B contiendra le numéro de famille de l'instruction.
210-220 : élimine le code « famille ». Garde l'instruction dans IN$.
La suite du listing représente le traitement particulier en fonction du type d'instruction. Noter le GOSUB calculé en fonction de B, numéro de famille, permettant l'aiguillage (ligne 260).
Ligne 265 : arrêt momentané pendant action sur SPACE.
3000 : partie DUMP. A1 et A2 sont les limites.
3020 : interruption momentanée du listage.
3040-3100 : on écrit 8 caractères sur la même ligne (ou un point pour remplacer caractères non imprimables).
PRINT CHR$ (17) supprime le curseur. Ceci est indispensable avec le système d'impression retenu (POKE du numéro de colonne), faute de quoi un curseur fixe resterait imprimé après chaque adresse ou octet de l'instruction.
PRINT CHRS (6) élimine le clavier sonore. Ceci est utile pendant le maintien de la barre espace.

Le plus grand soin doit être apporté lors de la recopie des lignes de DATA. Il ne faut pas sauter une virgule ou oublier un espace, chaque caractère ayant son importance.

Noter enfin que, pour l'adressage indexé, la virgule précédant le registre d'index a été remplacée par un POINT.

Vous disposez donc d'un programme permettant de désassembler la ROM de l'ORIC. Si vous désirez désassembler une routine en RAM, elle devra avoir été introduite avant le chargement du programme désassembleur. L'opération inverse reste néanmoins possible en prenant soin de modifier le sommet de la mémoire, par HIMEM, avant le chargement du désassembleur. La zone mémoire contenant la routine à désassembler sera ensuite chargée par CLOAD suivi des adresses de DÉBUT et de FIN.

```
 1  REM *****  DESASSEMBLEUR  *****
 2  REM *                           *
 3  REM *  D.BONOMO  &  E.DUTERTRE  *
 4  REM *                           *
 5  REM *     ORIC-1  &  ATMOS      *
 6  REM *                           *
 7  REM *       01-03-84  (V02)     *
 8  REM *                           *
 9  REM *****************************
```

```
100 DIMDA$(255)
120 FORI=0TO#FF:READDA$(I):NEXT
130 PAPER0:INK1
140 CLS
150 INPUT"1 - POUR DUMP   2 - POUR DESASSEMBLER ";CX
155 PRINT
160 ONCXGOTO3000,190
190 INPUT"ADRESSE DE DESASSEMBLAGE ";N,N2
192 PRINTCHR$(17);CHR$(6)
195 REPEAT
200 D$=DA$(PEEK(N))
202 PRINTHEX$(N);
205 B=VAL(D$):IFB=0THEN260
210 L=LEN(D$)-1
220 IN$=RIGHT$(D$,L)
230 POKE#269,9
235 PRINTHEX$(PEEK(N));
240 IFB<4THEN260
245 POKE#269,13
250 PRINTHEX$(PEEK(N+1));
255 POKE#269,22
260 GOSUB1000+B*100
265 IFPEEK(#208)=#38THEN270ELSEWAIT100
268 IFPEEK(#208)<>#38THEN269ELSE268
269 WAIT100
270 UNTILN>N2
280 PRINTCHR$(17);CHR$(6):PRINT:PRINT
290 GOTO150
999 REM---- PARTIE DESASSEMBLEUR ----
1000 REM Classe 0 - Inconnus
1015 POKE#269,35
1020 IF(PEEK(N))>=#20AND(PEEK(N)<127THENPRINTCHR$(PEEK(N))ELSEPRINT"."
1030 N=N+1
1099 RETURN
1100 REM Classe 1 - Implicite
1115 POKE#269,22
```

```
1120 PRINTIN$
1130 N=N+1
1199 RETURN
1200 REM Classe 2 - Addr 16 bits
1215 POKE#269,13
1220 PRINTHEX$(PEEK(N+1));
1225 POKE#269,17
1230 PRINTHEX$(PEEK(N+2));
1235 POKE#269,22
1240 PRINTIN$;"  ";
1260 PRINTHEX$(DEEK(N+1))
1270 N=N+3
1299 RETURN
1300 REM Classe 3 - AB,X AB,Y
1315 POKE#269,13
1320 PRINTHEX$(PEEK(N+1));
1330 POKE#269,17
1335 PRINTHEX$(PEEK(N+2));
1340 POKE#269,22
1345 PRINTIN$;
1350 POKE#269,26
1360 PRINTHEX$(DEEK(N+1))
1370 N=N+3
1399 RETURN
1400 REM Classe 4 - Donnee Immediate
1440 PRINTIN$;" %";HEX$(PEEK(N+1))
1450 N=N+2
1499 RETURN
1500 REM Classe 5 - Sauts
1540 PRINTIN$;"  ";
1545 IF(PEEK(N+1))>127THEN1560
1550 S=(N+2)+PEEK(N+1)
1555 GOTO1575
1560 S=(N+2)-(256-PEEK(N+1))
1575 PRINTHEX$(S)
1580 N=N+2
1599 RETURN
1600 REM Classe 6 - IND X
1640 PRINTIN$;" (   ,X)";
1645 POKE#269,27
1650 PRINTHEX$(PEEK(N+1))
1660 N=N+2
1699 RETURN
1700 REM Classe 7 - IND Y
1740 PRINTIN$;" (   ),Y";
1750 POKE#269,27
1760 PRINTHEX$(PEEK(N+1))
1780 N=N+2
```

```
1799 RETURN
1800 REM Classe 8 - Indexe Page zero
1835 PRINTIN$;
1840 POKE#269,26
1850 PRINTHEX$(PEEK(N+1))
1860 N=N+2
1899 RETURN
1900 REM Classe 9 - Addr 8 bits
1940 PRINTIN$;"  ";
1960 PRINTHEX$(PEEK(N+1))
1970 N=N+2
1999 RETURN
2999 REM ------ POUR LE DUMP ------
3000 INPUT"ADRESSES ZONE A DUMPER ";A1,A2
3005 PRINT:PRINT:PRINTCHR$(17);CHR$(6)
3010 FORI=A1TOA2STEP8
3020 IFPEEK(#208)=#38THEN3030ELSEWAIT100
3025 IFPEEK(#208)<>#38THEN3027ELSE3025
3027 WAIT20
3030 PRINTHEX$(I);:POKE#269,13
3040 FORJ=0TO7
3050 OC=PEEK(I+J)
3060 IFOC>#1FANDOC<#80THEN3080
3070 PRINT". ";:GOTO3100
3080 PRINTCHR$(OC);" ";
3100 NEXTJ
3200 PRINT
3210 NEXTI
3215 PRINT:PRINT
3220 PRINTCHR$(6);CHR$(17)
3300 GOTO150
4999 REM ---- TABLE DE DECODAGE ----
5000 DATA1BRK,6ORA,,,,9ORA,9ASL,,1PHP,4ORA,1ASL,,,2ORA,2ASL,
5005 DATA5BPL,7ORA,,,,8ORA     .X,8ASL     .X,,1CLC,3ORA       .Y,,,
5010 DATA3ORA       .X,3ASL       .X,,2JSR,6AND,,,9BIT,9AND,9ROL,,1PLP
5015 DATA4AND,1ROL,,2BIT,2AND,2ROL,,5BMI,7AND,,,,8AND     .X,8ROL     .X
5020 DATA,1SEC,3AND       .Y,,,,3AND       .X,3ROL       .X,,1RTI
5025 DATA6EOR,,,,9EOR,9LSR,,1PHA,4EOR,1LSR,,2JMP,2EOR,2LSR,
5030 DATA5BVC,7EOR,,,,8EOR     .X,8LSR     .X,,1CLI,3EOR       .Y,,,
5035 DATA3EOR       .X,3LSR       .X,,1RTS,6ADC,,,,9ADC,9ROR,
5040 DATA1PLA,4ADC,1ROR,,2JMP,2ADC,2ROR,,5BVS,7ADC,,,,8ADC     .X
5045 DATA8ROR     .X,,1SEI,3ADC       .Y,,,,3ADC       .X,3ROR       .X,,
5050 DATA6STA,,,9STY,9STA,9STX,,1DEY,,1TXA,,2STY,2STA,2STX,
5055 DATA5BCC,7STA,,,8STY     .X,8STA     .X,8STX     .Y,,1TYA
5060 DATA3STA       .Y,1TXS,,,3STA       .X,,,4LDY,6LDA,4LDX,
5065 DATA9LDY,9LDA,9LDX,,1TAY,4LDA,1TAX,,2LDY,2LDA,2LDX,
5070 DATA5BCS,7LDA,,,8LDY     .X,8LDA     .X,8LDX     .Y,,1CLV
```

```
5075 DATA3LDA       .Y,1TSX,,3LDY       .X,3LDA       .X,3LDX       .Y,
5080 DATA4CPY,6CMP,,,9CPY,9CMP,9DEC,,1INY,4CMP,1DEX,,2CPY,2CMP,2DEC,
5085 DATA5BNE,7CMP,,,,8CMP     .X,8DEC     .X,,1CLD,3CMP       .Y,,,
5090 DATA3CMP       .X,3DEC       .X,,4CPX,6SBC,,,9CPX,9SBC,9INC,
5095 DATA1INX,4SBC,1NOP,,2CPX,2SBC,2INC,,5BEQ,7SBC,,,
5100 DATA8SBC     .X,8INC     .X,,1SED,3SBC       .Y,,,,3SBC       .X
5105 DATA3INC       .X,
```

PROGRAMME DE CALCULS D'ATTÉNUATEURS EN PI OU EN TÉ

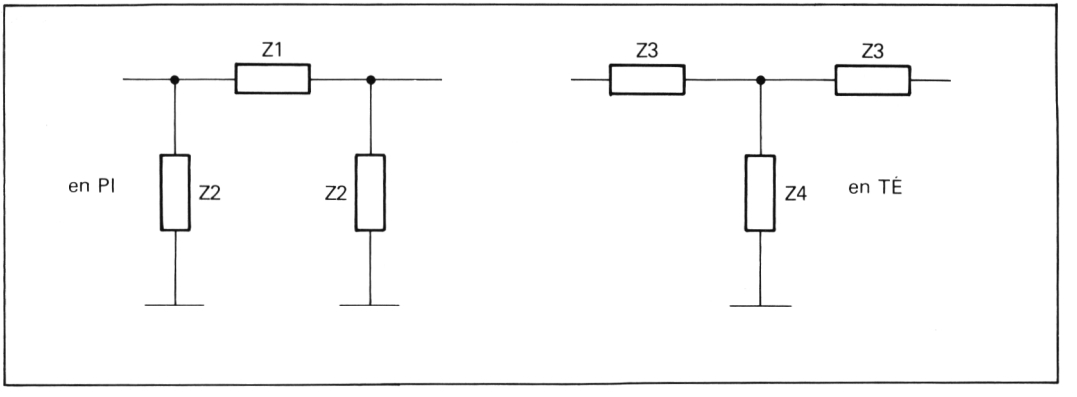
Il est souvent nécessaire, en communication, de disposer d'atténuateurs, pour rendre des signaux compatibles entre eux. Atténuateurs à l'entrée d'un récepteur, pour faire des mesures, ou entre un émetteur et un amplificateur, pour adapter les niveaux.

Le calcul d'un atténuateur en fonction de l'impédance caractéristique désirée, est chose facile lorsqu'on est muni d'un micro-ordinateur. Le programme proposé ici apporte la solution au problème tout en proposant le dessin des atténuateurs en PI et en TÉ, à l'écran.

Si on veut se passer de cette fonction non essentielle, alors on peut ne conserver que les lignes 1000 à 1180 (en ayant soin de supprimer la ligne 1010). En ligne 1190 mettre alors :

1190 PRINT Z1, Z2, Z3, Z4

pour obtenir les résultats, sachant que ces valeurs désignent les éléments recherchés, selon les schémas ci-dessous :



Si vous optez pour la solution « graphique », vous verrez les deux schémas se tracer à l'écran avec, en regard des composants, les valeurs correspondantes.

Le programme en lui-même n'appelle que peu de commentaires particuliers, si ce n'est l'utilisation de caractères redéfinis.

Pour dessiner les symboles de résistances à l'écran, tout en restant en mode TEXTE, nous avons utilisé la possibilité offerte par l'ORIC de redéfinir tout ou partie de son jeu de caractères. En effet, à l'initialisation de la machine, il y a transfert de la ROM vers la RAM de tout le jeu de caractères. Un coup d'œil sur la notice de la machine nous apprend que le premier caractère (l'espace, code 32) commence à l'adresse B500. Chaque caractère est codé sur 8 octets dont seuls 6 bits sont utilisés. (40 colonnes × 6 bits = 240 points adressables en horizontal et 24 lignes × 8 bits = 192 points en vertical, plus 3 lignes en mode « texte » ; on retrouve bien les valeurs du mode HIRES).

Nous savons retrouver l'adresse de la première ligne de la matrice de points d'un caractère donné, en fonction de son code ASCII. Il est donc aisé de redéfinir cette matrice, ce que nous avons fait ici avec les minuscules (a à k) qui sont remplacées par les « morceaux de résistances »...

Cette redéfinition des caractères se fait en 1500-1530 du programme, au moyen des profils binaires rangés en DATA, aux lignes 1600 à 1650.

La tabulation horizontale et le PRINT AT étant difficiles à utiliser, nous nous sommes servi des variables système en # 268 et # 269 pour accéder aux lignes et colonnes de l'écran devant recevoir un caractère.

Rappelons enfin que PRINT CHR$ (17) sert à supprimer (ou remettre) le curseur, ce qui est indispensable quand on imprime en utilisant # 268 et # 269.

```
1 REM ***** ATTENUATEURS *****
2 REM *    @ Eddy DUTERTRE     *
3 REM *    @ Denis BONOMO      *
4 REM *    F1EZH     F6GKQ     *
5 REM *     26-11-1983 V.01    *
6 REM *       O R I C - 1      *
7 REM *                        *
8 REM **************************
9 REM
1000 REM---- ATTENUATEURS PI/TE ----
1010 GOSUB1500
1020 CLS:PRINT:PRINT:PRINT
1025 INPUT"ATTENUATON DESIREE (DB) ";A
1030 N=EXP((A*2.302585/20))
1035 PRINT:PRINT
1040 INPUT"IMPEDANCE CARACTERISTIQUE (OHMS) ";ZC
1045 PRINT:PRINT
1100 Z1=ZC*((N*N-1)/(2*N))
1110 Z2=ZC*((N+1)/(N-1))
```

```
1120 Z1=INT(Z1*10)/10
1130 Z2=INT(Z2*10)/10
1150 Z3=ZC*((N-1)/(N+1))
1160 Z4=ZC*((2*N)/((N*N)-1))
1170 Z3=INT(Z3*10)/10
1180 Z4=INT(Z4*10)/10
1190 GOSUB1300
1200 PRINTCHR$(17)
1205 POKE#268,15:PRINT:POKE#269,8:PRINTZ1;
1210 POKE#269,26:PRINTZ3
1230 POKE#268,20:PRINT
1240 POKE#269,8:PRINTZ2;
1250 POKE#269,30:PRINTZ4
1260 POKE#268,25:PRINT
1265 PRINT"ATTENUATION DE ";A;" DB SOUS ";ZC;" OHMS"
1270 PRINTCHR$(17):PRINT"DESIREZ-VOUS UN AUTRE CALCUL (O/N)":GETR$
1275 IFR$="O"THEN1020ELSEEND
1300 REM---- DESSINE ----
1310 POKE#268,17:PRINT
1320 PRINT"    ";"a";"i";"i";"f";"j";"9";"i";"i";"a"
1330 PRINT"    ";"b";"       ";"b"
1340 PRINT"    ";"d";"       ";"d"
1350 PRINT"    ";"k";"       ";"k"
1360 PRINT"    ";"e";"       ";"e"
1370 PRINT"    ";"b";"       ";"b"
1380 PRINT"    ";"c";"       ";"c"
1390 POKE#268,17:PRINT
1400 POKE#269,23:PRINT"h";"f";"j";"9";"h";"a";"h";"f";"j";"9";"h"
1405 POKE#269,28:PRINT"b"
1410 POKE#269,28:PRINT"d"
1420 POKE#269,28:PRINT"k"
1430 POKE#269,28:PRINT"e"
1440 POKE#269,28:PRINT"b"
1450 POKE#269,28:PRINT"c"
1470 RETURN
1500 FORA=46856TO46943
1510 READD$:D=VAL("#"+D$)
1520 POKEA,D:NEXT
1530 RETURN
1600 DATA0C,0C,1E,3F,3F,1E,0C,0C
1605 DATA0C,0C,0C,0C,0C,0C,0C,0C
1610 DATA0C,0C,0C,0C,0C,0C,3F,3F
1615 DATA3F,21,21,21,21,21,21,21
```

```
1620 DATA21,21,21,21,21,21,21,3F
1625 DATA00,3F,20,20,20,20,3F,00
1630 DATA00,3F,01,01,01,01,3F,00
1635 DATA00,00,00,3F,3F,00,00,00
1640 DATA00,00,00,3F,3F,00,00,00
1645 DATA00,3F,00,00,00,00,3F,00
1650 DATA21,21,21,21,21,21,21,21
```

FRÉQUENCE

Le programme « FRÉQUENCE » va nous permettre d'utiliser ORIC-1 d'une manière un peu particulière. En effet, nous allons grâce à lui, mesurer une fréquence qu'il sera possible de générer, ensuite, avec le circuit sonore. Nous verrons dans quelles limites ce programme est utilisable.

Pour mesurer la fréquence, nous n'utiliserons aucun circuit extérieur et introduirons le signal à mesurer sur la prise reliée normalement à la sortie du magnétophone. La structure du circuit d'entrée de l'ORIC le permet puisque les signaux sinusoïdaux qui y sont introduits sont amplifiés et rendus compatibles TTL avant d'être envoyés sur le VIA interne.

Nous programmerons ce même VIA pour utiliser un TIMER qui nous servira de « base de temps » pour la mesure.

Le principe de la mesure est simple : on compte les impulsions reçues par la broche CB1 du VIA, pendant un intervalle de temps calibré par le TIMER 2.

Les lecteurs qui désirent saisir parfaitement le fonctionnement de ce programme doivent avoir bien assimilé la programmation du VIA 6522. Partant de cet acquit, nous ne développerons que les grandes lignes.

Le temps de base chargé dans le timer 2 est 65 ms (le TIMER décompte sur réception de l'horloge de base, de période 1 μs, ce qui permet de compter, sur 16 bits, jusqu'à 65535. Nous avons choisi la valeur pratique de 65000).

Pour obtenir une précision acceptable, nous répéterons l'opération 10 fois. Le temps total de la mesure sera de 650 ms. Les impulsions reçues pendant le temps de comptage de base (65 ms) sont comptées dans un registre. Il ne pourra en recevoir au maximum que 255 et au minimum... 1 seule. Voilà donc fixées les limites de notre mesure : 10 impulsions pendant 0.65 s = environ 15 Hz et 2550 impulsions pendant 0.65 s = environ 3900 Hz.

Le tableau présenté en annexe permet de constater quel est l'effet d'une variation du temps de comptage sur les limites inférieure et supérieure ainsi que sur le « pas » de la mesure. On a baptisé « pas » l'écart de fréquence (qui résulte du calcul) entre deux mesures comptant une impulsion de différence. Ce pas existe car le registre compteur ne peut emmagasiner que des valeurs entières.

Nous vous invitons à vous référer à l'organigramme de principe pour comprendre le fonctionnement du programme.

Le VIA est programmé, à l'origine dans l'ORIC, pour compter les transitions positives de la ligne CB1.

Pour améliorer la précision il est indispensable d'inhiber les interruptions pendant le comptage.

Rappelons que le TIMER est lancé quand on charge son octet de poids fort, qu'il fonctionne en décompteur et qu'il positionne un bit du registre indicateur d'interruptions (IFR) du VIA quand il passe par zéro.

L'indicateur d'interruption de CB1 est effacé par lecture du Registre de sortie du PORT B. celui du TIMER est effacé lors de son chargement.

ADRESSES DES REGISTRES DU VIA UTILISÉS

0300	sortie PORT B	
0308	octet poids faible	TIMER 2
0309	octet poids fort	
030D	registre indicateur d'interruptions (IFR)	
	Bit 4 : transition CB1	
	Bit 5 : passage à zéro du TIMER 2.	

ADRESSES UTILISÉES PAR LE PROGRAMME

0400	octet de poids faible	valeur chargée dans TIMER 2
0401	octet de poids fort	
0402	compteur du nombre de mesures	
0410/041A	résultats des mesures	

INCIDENCE DU TEMPS DE LA MESURE SUR SON RÉSULTAT

Temps de la Mesure	0,65	0,5	0,3	0,1	
Nb. d'impulsions — Fréquence					
10	15	20	33	100	Fréquence mini-mesurable
11	17	22	37	110	
12	18	24	40	120	
299	460	598	997	2990	
300	462	600	1000	3000	
301	463	602	1003	3010	
2548	3920	5096	8493	25480	
2549	3922	5098	8497	25490	
2550	3923	5100	8500	25500	Fréquence maxi-mesurable

INCIDENCE SUR LA MESURE D'UN 1750 Hz

0,65 s		0,5 s		0,3 s		0,1	
fréq.	nb. impuls.	F.	nb. imp.	F.	nb. imp.	F.	nb. imp.
1748	1136	1746	873	1743	523	1730	173
1749	1137	1748	874	1747	524	1740	174
1751	1138	1750	875	1750	525	1750	175
1752	1139	1752	876	1753	526	1760	176

QUELQUES COMMENTAIRES SUR LE LISTING

Nous avons déjà indiqué que le programme effectue 10 mesures successives. Les résultats de ces 10 mesures sont rangés entre 041A et 0411.

Le compteur de mesures est initialisé en 0402. Le TIMER 2 est initailisé à partir des adressses 0400 et 0401.

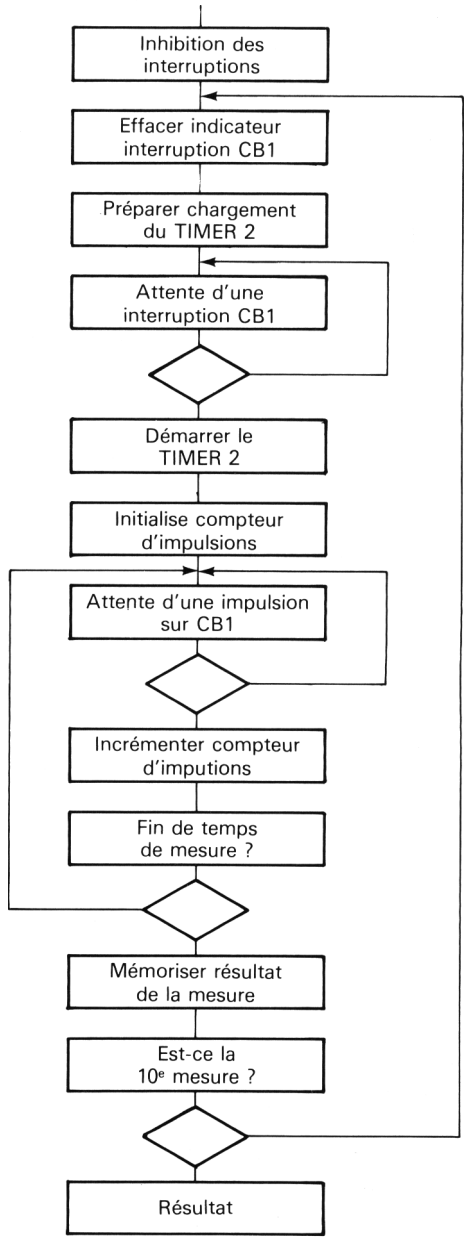
La fréquence mesurée est arrondie à la valeur entière la plus proche. Cette valeur de fréquence peut être envoyée, ou non, au générateur sonore, suivant le choix fait par l'opérateur.

Les lignes de DATA contenant les octets du langage machine sont présentées groupées par instructions. Cette présentation n'est, bien sûr, pas obligatoire. Elle est concevable pour un programme court, comme celui-ci, et favorise la lecture.

Lignes 10 à 50 : implantation du langage machine
90 à 95 : initialisation des valeurs TIMER et compteur mesures.
96 à 98 : Bascule marche arrêt du générateur sonore.
1 000 à 2 000 : instructions du langage machine.
3003 : on relit les valeurs des 10 mesures
3005 : calcule la fréquence après avoir fait la moyenne.
3007 : arrondit la valeur de la fréquence.
3010-3015 : affiche la valeur par la fonction STR$ après traitement pour supprimer le caractère parasite ajouté par le BASIC de l'ORIC.
3020-3030 : envoie la valeur au générateur sonore si ce choix a été fait.

Le mode d'emploi n'appelle aucun commentaire puisqu'il n'y a besoin que d'envoyer le signal dont on veut mesurer la fréquence sur la prise reliée, normalement à l'écouteur du magnétophone.

ORGANIGRAMME DE FONCTIONNEMENT



```
0 REM ***** FREQUENCE *****
1 REM *                     *
2 REM *D.BONOMO E.DUTERTRE*
3 REM * F6GKQ        F1EZH *
4 REM *       19-10-83      *
5 REM *    O R I C - 1      *
6 REM *        V.01         *
7 REM ***********************
8 REM
9 PRINTCHR$(6)
10 HIMEM#9600
20 REPEAT
30 READD:POKE(#9700+I),D
40 I=I+1
50 UNTILD=#88
60 X=10:Y=10
70 CLS
75 S=0
80 PLOTX-4,Y-2,"FREQUENCE (HERTZ)"
85 PLOTX-4,Y+10,"ECOUTE:TOUCHE 'S'"
90 POKE#400,232:POKE#401,253
95 POKE#402,10
96 T$=KEY$
97 IFT$="S"ANDS=1THENS=0:GOTO100
98 IFT$="S"ANDS=0THENS=1
100 CALL#9700:GOSUB3000:GOTO95
1000 DATA#78
1005 DATA#AE,#00,#04
1010 DATA#8E,#08,#03
1015 DATA#AE,#01,#04
1020 DATA#AD,#00,#03
1025 DATA#A9,#10
1030 DATA#2C,#0D,#03
1035 DATA#F0,#FB
1040 DATA#8E,#09,#03
1045 DATA#AE,#00,#03
1050 DATA#A0,#00
1055 DATA#AA
1060 DATA#8A
1065 DATA#2C,#0D,#03
1070 DATA#F0,#FB
1075 DATA#AE,#00,#03
1080 DATA#C8
```

```
1085 DATA#AA
1090 DATA#A9,#20
1095 DATA#2C,#0D,#03
1100 DATA#F0,#EE
1105 DATA#AE,#02,#04
1110 DATA#F0,#0B
1113 DATA#98
1115 DATA#9D,#10,#04
1120 DATA#CA
1125 DATA#8E,#02,#04
1130 DATA#4C,#01,#97
1135 DATA#58
1140 DATA#60
2000 DATA#88
3000 REM---- CALCULE LA FREQ. -----
3002 I=0
3003 FORN=0TO9:I=I+(PEEK(#041A-N)):NEXT
3005 FCE=I/65E-2
3007 IF(FCE-INT(FCE))>=.5THENFCE=1+INT(FCE)ELSEFCE=INT(FCE)
3010 FCE$=MID$(STR$(FCE),2)+"    "
3015 PLOTX,Y,FCE$
3020 IFS=1THEN SOUND1,(62500/FCE),5
3030 IFS=0THENSOUND1,100,0
3040 RETURN
```

```
01
9700 78      SEI
9701 AE0004 LDX #0400
9704 8E0803 STX #0308
9707 AE0104 LDX #0401
970A AD0003 LDA #0300
970D A910   LDA %#10
970F 2C0D03 BIT #030D
9712 F0FB   BEQ #970F
9714 8E0903 STX #0309
9717 AE0003 LDX #0300
971A A000   LDY %#00
971C AA     TAX
971D 8A     TXA
971E 2C0D03 BIT #030D
9721 F0FB   BEQ #971E
9723 AE0003 LDX #0300
```

inhibition des interruptions
chargement de l'octet de poids faible du TIMER 2

efface l'indicateur d'interruption sur CB1
attend un front montant (interruption) sur CB1

lance le TIMER2
Efface indicateur d'interruption sur CB1
initialise le compteur d'impulsions

Attente d'une autre impulsion

Efface indicateur d'interruption sur CB1
incrémente le compteur d'impulsions

```
9726 C8      INY
9727 AA      TAX
9728 A920    LDA %#20      Teste si TIMER 2 à zéro (interruptions)
972A 2C0D03  BIT #030D
972D F0EE    BEQ #971D     si non, continue mesure
972F AE0204  LDX #0402     compteur de mesures à la dixième mesure ?
9732 F00B    BEQ #973F     si ou on arrête
9734 98      TYA
9735 9D1004  STA #0410,X   Range le résultat
9738 CA      DEX           Décrémente et sauvegarde le compteur de
9739 8E0204  STX #0402     mesures
973C 4C0197  JMP #9701     Recommence
973F 58      CLI           Autorise les interruptions pour le retour
9740 60      RTS           du BASIC
```

PROGRAMME MIRE

Pour exploiter les possibilités couleurs de l'ORIC nous avons choisi de le coupler à l'émetteur de télévision d'amateur de la station. En effet, il est permis, grâce à l'ORIC, de tracer à l'Ecran une mire en couleurs dont la présentation n'est pas sans rappeler celles que nous voyons quotidiennement sur les trois chaînes nationales, mais de format rectangulaire.

Cette mire pourra être utilisée à d'autres fins, et notamment au réglage d'un téléviseur couleur. Il est à noter que les couleurs produites par l'ORIC sont de très belle qualité.

Une autre caractéristique de l'ORIC étant de posséder un générateur de caractères entièrement redéfinissable par l'utilisateur, puisqu'implanté en RAM (mémoire vive), nous ne nous sommes pas privé d'y faire appel pour créer les caractères nécessaires (essentiellement des groupes de barres plus ou moins proches).

Le principe pourra être retenu sur toute autre machine permettant les mêmes fantaisies.

Le caractère ainsi redéfini est imprimé à l'écran, soit par la fonction CHR$, soit en l'appelant sous son ancienne forme. En effet, nous avons redéfini les caractères inutilisés, tels que les minuscules. Vous trouverez donc, dans le programme les deux formes suivantes : PRINT CHR$ (109)

ou PRINT« m ».

Outre ses dessins de barres et sa palette de couleurs, cette mire vous donnera l'heure avec une précision très acceptable. Pour ce faire, la procédure de mise à l'heure est la suivante. Entrer l'heure lorsque ORIC vous le demande (2 caractères tels que : 09).

Entrer les minutes en prévoyant, par exemple, une minute d'avance (2 caractères tels que : 53).

La mire se dessine alors à l'écran. La machine attend maintenant que vous appuyiez sur une touche, ce que vous ferez au top horaire de l'horloge parlante, initialisant ainsi le comptage.

L'heure est affichée dans le haut de la mire, l'indicatif de la station utilisatrice (ou tout autre texte) dans le bas.

Rappelons simplement que l'ORIC possède des fonctions d'affichage en double hauteur, en fixe ou clignotant. C'est une des remarquables possibilités de la machine, qui permet de mélanger sur un même écran des textes en double hauteur, simple hauteur certains restant fixes ou d'autres clignotant...

Pour obtenir ces diverses options on dispose de caractères de contrôle. Le code ESCape, CHR$ (27), permet d'y accéder.

La syntaxe est un peu particulière puisque ce code est suivi d'un second provocant l'effet voulu.

Ainsi PRINT CHR$ (27) ; « N FGGKQ » fera apparaître l'affichage de l'indicatif en double hauteur et clignotant. Ceci est décidé par la lettre N devant le texte.

Ces particularités de gestion de l'affichage sont citées dans la notice d'exploitation de la machine, mais on ne peut pas dire que les explications soient généreuses !

Examinons maintenant le listing du programme où nous avons volontairement mélangé l'utilisation de PRINT CHR$ ( ) et de PRINT « caractère ».

Ligne 40 : On efface dans un but d'esthétique, l'inscription « CAPS », indiquant le mode majuscules, située normalement à droite de la première ligne de l'écran, aux adresses 48036 à 48039 (hexadécimal BBA4 à BBA9), en affichant à la place, des blancs (code 32 en décimal). Cette opération ne pouvant être faite par PRINT, est faite par des POKE.

Notez que, dans les boucles FOR-NEXT, lorsqu'il n'y a pas de risque d'ambiguïté, il est possible d'omettre la variable derrière NEXT, ce qui provoque un gain de vitesse à l'exécution. Ceci n'a pas été fait ici, dans un souci de clarté du programme.

Lignes 50-95 : Introduction de l'heure pour initialiser la pendule. Respecter la procédure expliquée plus haut.

Lignes 100-125 : On définit l'ordre des couleurs, dont les codes sont rangés dans la table CL, pour que leur présentation à l'écran soit harmonieuse et provoque sur un téléviseur Noir et Blanc, une impression de dégradé de gris. (Beaucoup d'amateurs ATV ne sont encore équipés q'en Noir et Blanc.)

Ligne 220 : A$ est initialisée avec le code de ESCape C$ est initialisée avec le code de Double Hauteur. Ceci nous permettra par la suite de ne plus avoir à écrire, à chaque utilisation CHR$ (27) ou CHR$(4) mais simplement A$ et C$.

Lignes 250 à 300 : C'est à cet endroit du programme que les caractères spéciaux, et symboles utilisés dans la mire, sont redéfinis.

Pour les lecteurs utilisant une autre machine, signalons simplement que le matriçage d'un caractère est de 6 colonnes × 8 lignes sur ORIC. La « valeur » de chaque ligne étant donnée par son profil binaire, les deux bits de poids fort forcés à zéro.

Lignes 1000 à 1750 : On trace les différents secteurs de la mire en utilisant les caractères redéfinis précédemment.

Ligne 1720 : Elle contient l'indicatif. Vous la modifierez donc à votre goût, pour personnaliser votre mire. Sachez simplement qu'il vous faudra respecter le nombre de caractères situés après le N (lettres chiffres ou blancs), faute de quoi le reste de la mire serait décalé d'autant.

Ligne 1790 : La mire a été tracée. On attend l'appui sur une touche, en coïncidence avec un top de l'horloge parlante, pour démarrer la pendule.

Ligne 1795 : On efface le curseur clignotant. Pour d'autres machines, utiliser le caractère de servitude approprié.

Lignes 1800-1895 : Affichage de l'heure.
Les variables HH (pour l'heure) et MM (pour les minutes) sont incrémentées en fonction du temps. Elles sont ensuite transformées, grâce à la fonction STR$, en chaînes de caractères qui seront affichées, par la fonction PLOT, en haut de la mire.

Ligne 1850 : Le WAIT 99 ajuste la précision dù comptage.

Suivent maintenant quelques explications pour le couplage de l'ORIC à un émetteur de télévision d'amateur, ou à toute autre entrée vidéo.

Nous avons déjà écrit que les sorties RVB de l'ORIC pouvaient sans danger être reliées entre elles et mélangées à la SYNCHRO. Cet artifice permet de disposer d'une sortie VIDÉO NOIR et BLANC.

Le signal « composite » est alors d'environ 800 mV chargé par 75 Ω de l'entrée vidéo d'un émetteur de télévision.

Pour une utilisation en couleurs, il reste 2 possibilités :

1. Entrer les 4 signaux R, V, B et SYNCHRO sur un codeur SECAM.
2. Prendre la vidéo couleur (standard PAL) disponible à l'intérieur de l'ORIC, à l'entrée du modulateur UHF (ou au point commun R15 et RV1).

```
1 REM   +++++ MIRE +++++
2 REM   + @ E. DUTERTRE+
3 REM   + @ D. BONOMO  +
4 REM   + F1EZH-F6GK0  +
5 REM   +  25-08-1983  +
6 REM   +    (V.01)    +
7 REM   +  O R I C - 1 +
8 REM   +              +
9 REM   ++++++++++++++++
10 CLS
40 FORI=48036TO48039:POKEI,32:NEXTI
45 PRINT
50 PRINT"MISE A L'HEURE DE L'HORLOGE"
55 PRINT"----------------------------"
60 PRINT
65 PRINT"INTRODUIRE LES HEURES ET LES MINUTES"
70 PRINT"AVEC UN PEU D'AVANCE SUR LE TOP "
75 PRINT
80 PRINT"APPUYEZ SUR UNE TOUCHE AU TOP"
85 PRINT
90 INPUT"     LES HEURES :";HH
95 INPUT"     LES MINUTES:";MM
97 CLS
100 DIMCL(8)
200 FORI=1TO8
205 READCL(I)
210 NEXTI
215 DATA144,148,145,149,146,150,147,151
28 A$=CHR$(27):C$=CHR$(4)
250 FORI=46856TO46983
255 READDI
260 POKEI,DI
265 NEXTI
270 DATA1,1,1,1,1,1,1,1
272 DATA32,32,32,32,32,32,32,32
274 DATA63,63,0,0,0,0,0,0
276 DATA0,0,0,0,0,0,63,63
278 DATA63,63,1,1,1,1,1,1
280 DATA63,63,32,32,32,32,32,32
282 DATA63,63,12,12,12,12,63,63
284 DATA63,63,32,32,32,32,63,63
286 DATA63,63,1,1,1,1,63,63
288 DATA63,63,63,0,0,63,63,63
```

```
290 DATA7,7,7,7,7,7,7,7
292 DATA27,27,27,27,27,27,27,27
294 DATA21,21,21,21,21,21,21,21
296 DATA63,63,63,63,63,63,63,63
298 DATA32,32,32,32,32,32,63,63
300 DATA1,1,1,1,1,1,63,63
1000 PRINT"  ";
1005 FORI=1TO33:PRINTCHR$(100);:NEXTI
1010 PRINT"  "
1100 PRINT"  ";C$;CHR$(98);"
1101 PRINTA$;"P";A$;"G";A$;"J";
1102 PRINT"HH:MM";"    ";
1103 PRINTA$;"W";A$;"@";"
1104 PRINTA$;"J";
1105 PRINTCHR$(97)
1110 PRINTC$
1190 FORJ=1TO2
1200 PRINT"  ";
1205 FORI=1TO3:PRINTCHR$(255);:NEXTI

1210 PRINT" ";CHR$(97);
1215 PRINT"
1220 FORI=1TO3:PRINTCHR$(255);:NEXTI
1230 PRINT
1240 NEXTJ
1250 FORJ=1TO2
1255 PRINT"  ";CHR$(255);
1260 FORI=1TO8:PRINTCHR$(254);CHR$(254);CHR$(255);CHR$(255);:NEXTI
1270 PRINT
1275 NEXTJ
1300 FORI=1TO5
1305 PRINT"  ";
1310 FORJ=1TO8
1320 PRINTA$;CHR$(CL(J));"    ";
1330 NEXTJ
1340 PRINTA$;CHR$(151);
1345 PRINTCHR$(98)
1350 NEXTI
1390 PRINT"  ";CHR$(104);
1400 FORI=1TO31:PRINTCHR$(103);:NEXTI
1410 PRINTCHR$(105);" "
1420 PRINT"  ";
1430 FORI=1TO33:PRINTCHR$(106);:NEXTI
```

```
1435 PRINT"   "
1450 PRINT"   ";CHR$(104);
1460 FORI=1TO31:PRINTCHR$(103);:NEXTI
1342 PRINTCHR$(105);"  "
1308 FORI=1TO5
1505 PRINT"   ";
1510 PRINTCHR$(255);CHR$(255);CHR$(255);
1515 PRINT"k";"k";"k";
1520 PRINT"l";"l";"l";"l";"l";
1525 PRINT"m";"m";"m";"m";"m";"m";"m";"m";"m";"m";"m";
1530 PRINT"l";"l";"l";"l";"l";
1535 PRINT"k";"k";"k";
1540 PRINTCHR$(255);CHR$(255);CHR$(255)
1545 NEXTI
1600 FORI=1TO3
1610 PRINT"   ";CHR$( 98);
1620 FORJ=8TO1STEP-1
1630 PRINTA$;CHR$(CL(J));"    ";
1640 NEXTJ
1645 PRINTA$;CHR$(151)
1650 NEXTI
1700 PRINTC$;
1710 PRINT"   ";CHR$( 98);"   ";
1715 PRINTCHR$(110);CHR$(110);CHR$(107);
1720 PRINTA$;"P";A$;"G";A$;"N F 6 G K 0 - 9 1     ";A$;"W";A$;"@";A
;"J";
1725 PRINTCHR$(97)
1730 PRINTC$
1740 PRINT"   ";"o";
1745 FORI=1TO31:PRINTCHR$(100);:NEXTI
1750 PRINT"P"
1790 GETK$
1795 PRINTCHR$(17);
1797 PLOT17,1,"     ":PLOT17,2,"     "
1800 MM$=STR$(MM)
1805 HH$=STR$(HH)
1810 IFHH>9THEN1818
1812 PLOT18,1,HH$:PLOT18,2,HH$
1814 PLOT18,1,"0":PLOT18,2,"0"
1816 GOTO1820
1818 PLOT17,1,HH$:PLOT17,2,HH$
1820 IFMM>9THEN1828
1822 PLOT21,1,MM$:PLOT21,2,MM$
```

```
1824 PLOT21,1,"0":PLOT21,2,"0"
1826 GOTO1845
1828 PLOT20,1,MM$:PLOT20,2,MM$
1845 PLOT20,1,":":PLOT20,2,":"
1850 FORI=1TO60:WAIT 99:NEXTI
1855 WAIT40
1860 MM=MM+1
1865 IFMM<60THEN1800
1870 MM=0
1875 HH=HH+1
1880 IFHH<24THEN1800
1890 HH=0
1895 GOTO1800
1955 PRINTCHR$(17);
1960 GETK$
1965 PRINTCHR$(17)
```

LOCATORIC

L'une des applications utiles et essentielles de l'ORIC, se situe dans le domaine du calcul. Nous proposons ici un programme d'application qui permet de déterminer les distances par le procédé dit de « QTH LOCATOR ».

Outre ses fonctions de calculs satisfaisantes, ORIC est doté de possibilités graphiques non négligeables, que nous avons tenté d'exploiter ici en partie, en dessinant la carte de France et en y faisant figurer les positions respectives des stations de départ et du correspondant. L'azimut entre les deux sera matérialisé par un trait, reliant ces deux points.

Pour rendre ce programme encore plus utile, nous y avons incorporé une fonction permettant de déterminer le QTH LOCATOR, en partant des coordonnées géographiques du lieu.

Ces coordonnées, qui peuvent être trouvées sur une carte, doivent être fournies à la machine sous forme de degrés décimaux, la longitude étant exprimée par rapport au méridien de Greenwich.

Le programme a été conçu de façon modulaire, ce qui permet de supprimer éventuellement une fonction ou d'y apporter toute modification. D'autre part, exclusion faite des fonctions graphiques qui demanderont d'éventuelles adaptations, le programme pourra fonctionner sur d'autres machines utilisant le même type de BASIC.

DESCRIPTION DÉTAILLÉE DU PROGRAMME

Lignes 10 à 25 on trouve le « menu » qui permet l'aiguillage entre les fonctions « calcul distance-azimut » ou « détermination du QTH locator ». Pour ce faire, on utilise l'ordre BASIC

ON [variable condition] GOTO [adresse 1, adresse 2] qui permet, selon le contenu de la variable (ici R) de dérouter le programme vers les adresses indiquées.

A — CAS DE LA DÉTERMINATION DU QTH-LOCATOR

Il suffit de se souvenir du principe de découpage de la grille QTH Locator, pour comprendre le principe de cette partie du programme.

Ainsi, les grands rectangles désignés par les 2 lettres « mesurent » 1° en écart de latitude et 2° en écart de longitude. Les petits rectangles, désignés par les 2 chiffres représentent un écart de 7'30'' en latitude et de 12' en longitude. Le petit rectangle, désigné par la dernière lettre du QTH Locator, représente 2'30'' d'écart en longitude et 4' en latitude.

Partant de ce principe, et en effectuant des divisions successives, on arrivera aisément au résultat.

Rappelons seulement que, dans un codage QTH Locator représenté par ABCDE, on a :

A fonction de la longitude
B fonction de la latitude
C fonction de la longitude
D fonction de la latitude
E fonction de la longitude et de la latitude.

On détermine ainsi les différents éléments A$, B$, C$, D$ et E$ qui composent la chaîne Q$ (ligne 6000) du QTH Locator.

Ainsi, la lettre I correspondant à 48° de latitude, se retrouve en utilisant son code ASCII (valeur 73) et en additionnant un « offset » de 25 (73 = 48 + 25). On retrouve ainsi toutes les lettres du codage latitude (Code ASCII = Latitude + 25).

La ligne 6120 montre cette transformation où, grâce à la latitude, on remonte au code ASCII puis au caractère grâce à la fonction CHR$ du BASIC. Le chiffre, correspondant au caractère C$ de la chaîne, se déduit ensuite aisément puisqu'il est fonction de la partie décimale de la latitude, et qu'il correspond à des incréments de 7'30'' (ou 7,5 minutes) ce qu'on retrouve à la ligne 6125. On arrive ainsi jusqu'à la petite lettre, subdivision du rectangle désigné par C$ et D$, et qui est codée dans le programme par la variable EL.

De 6300 à 6480, on procède de même pour la longitude, jusqu'à atteindre la variable EG. Le tableau T$ permet de retrouver la « petite lettre » du QTH Locator (E$) dans lequel on entre grâce aux variables EL EG, à la ligne 6500.

Dans cette partie du programme, nous n'avons pas utilisé de fonctions particulières.

B — PARTIE CALCUL DISTANCE-AZIMUT

Comme nous l'avons déjà signalé, le programme calcule la distance et l'azimut entre les 2 stations et dessine la carte de FRANCE, ce qui permet une meilleure localisation.

1. TRACÉ DE LA CARTE

La partie du programme qui trace les contours du pays est située entre les lignes 3999 et 5040. Le principe est simple :

a) On passe en haute résolution par l'ordre HIRES de l'ORIC ce qui a pour effet d'effacer l'écran et de laisser 3 lignes en mode Texte, en bas. Sur un autre ordinateur, vous utiliserez la fonction appropriée.

b) On sélectionne les couleurs du tracé (INK) et du fond (PAPER), désirées. La ligne 4005 n'intéresse donc pas ceux qui utilisent un moniteur Noir et blanc.

c) On positionne le curseur et on trace les contours du pays définis par 54 points (X, Y) qu'on relie entre eux.

Note : CURSET X, Y, a positionne le curseur aux endroits de l'écran (X de 0 à 239) (et Y de 0 à 199) désignés et allume le point correspondant si a = 1.

DRAW X, Y, a trace la ligne définie par la position précédente du curseur et par celle qui est évoquée par X et Y. La ligne est « allumée » sur l'écran si a = 1.

2. ALLUMAGE DU POINT CORRESPONDANT À LA STATION ORIGINE ET CALCUL DES COORDONNÉES DE LA STATION

a) Le calcul des coordonnées s'effectue dans le sous-programme implanté entre les lignes 70 et 390, détaillé plus loin.

b) Pour allumer le point correspondant à votre station, il faudra procéder par tâtonnements pour déterminer ses coordonnées X et Y. Méthode peu informatique, mais le sous-programme de calcul eût été un peu long. Ce point est défini par X0 et Y0 (ligne 450) et allumé par CURSET X0, Y0, 1 en ligne 460.

Procéder par essais successifs, en faisant Break après le tracé de la carte (on reste en HIRES) et en tapant CURSET X, Y, 1 où X et Y sont respectivement compris entre Ø et 239 et Ø et 199. Pour vous guider, le point cité dans le programme à la ligne 450 (et dont vous remplacerez les coordonnées par celles que vous aurez déterminé et pris soin de noter) correspond au QTH de l'auteur (CORBEIL, BI 23 avec XØ = 121 et YØ = 54).

c) Le tracé du parcours entre les 2 stations est effectué entre les lignes 750 et 870. Le but recherché n'était pas d'atteindre une grande précision et, pour calculer les coordonnées du point signalant la station du correspondant, on a considéré que la France était inscriptible dans un carré en ce qui concerne les écarts extrêmes entre longitude et latitude, d'ouest en est et du nord au sud.

Enfin, si le point à allumer est hors de l'écran il est évident qu'on ne le fera pas. C'est le rôle de la ligne 765.

3. TRANSFORMATION DES COORDONNÉES

Le principe utilisé s'avère assez précis et autorise le calcul de distances pour une zone couvrant l'Europe de l'Ouest.

Les fonctions de découpage de chaîne sont standard. On traite ainsi les 5 caractères du QTH Locator par leur code ASCII.

Ligne 70 : on traite différemment les Locators Ouest de Greenwich (code des lettres T à Z). Par ce principe, tous les locators dont les lettres commencent par A à S seront en longitudes positives, les autres en longitudes négatives. Les formules d'exploitation et de transformation des différents coefficients obtenus sont situées aux liGNES 330 et 340.

4. CALCUL DE LA DISTANCE ET DE L'AZIMUT

La formule utilisée n'est autre que celle employée couramment en navigation, qui permet de connaître la distance séparant 2 points du globe de coordonnées LB et GB et LA et GA, qui est en fait la longueur de l'arc de grand cercle dont le centre est celui de la terre et qui passe par ces 2 points. Cette formule est suffisamment précise si la route est courte.

Rappelons, qu'à l'équateur, un angle de 1 degré correspond à environ 111 kms (60 miles nautiques). 1 mile nautique égale à 1 minute d'angle.

Voici les formules :

$$\text{Dist} = \text{ARCOS}\,[\sin(LA)\sin(LB) + \cos(LA)\cos(LB)\cos(GB-GA)] * 60$$

$$\text{AZI} = \text{ARCOS}\,\frac{\sin(LB) - \cos(D/60)\sin(LA)}{\sin(D/60)\cos(LA)}$$

La distance est exprimée en kilomètres et l'azimut en degrés.

Le micro-ordinateur, travaillant toujours en modes « radians », il faut en tenir compte dans les calculs.

Ligne 630 ACSX représente l'arc cosinus qui n'est pas directement disponible sur l'ORIC et qu'on calcule à partir de l'arc-tangente (ATN).

On arrondit les résultats, aux lignes 645 et 735.

Ligne 850, on attend l'appui sur une touche pour effacer la ligne tracée, le bas de l'écran, et recommencer un nouveaul calcul.

```
1 REM    +++ LOCATORIC  +++
2 REM    + BONOMO DUTERTRE+
3 REM    + F6GKQ  F1EZH   +
4 REM    +   05-08-1983   +
5 REM    +     V.01       +
6 REM    +   O R I C - 1  +
7 REM    ++++++++++++++++++
8 REM
9 CLS:TEXT:PRINT:PRINT
10 PRINT" 1 -POUR CALCULER DES DISTANCES"
12 PRINT"    EN FONCTION DU QTH LOCATOR"
14 PRINT
15 PRINT" 2 -POUR DETERMINER UN LOCATOR"
17 PRINT"    EN FONCTION DES COORDONNEES"
19 PRINT
20 INPUT"VOTRE CHOIX ";R
22 CLS
25 ONRGOTO4000,6000
69 REM+++TRANSFO LOC COORDONNEES+++
70 IFASC(Q$)<84THENGOTO100
80 A=-91+ASC(Q$)
90 GOTO110
100 A=-65+ASC(Q$)
110 Q$=RIGHT$(Q$,4)
120 B=-65+ASC(Q$)
130 Q$=RIGHT$(Q$,3)
140 C=-48+ASC(Q$)
150 Q$=RIGHT$(Q$,2)
160 D=-48+ASC(Q$)
```

```
170 Q$=RIGHT$(Q$,1)
180 E=ASC(Q$)
190 IFD<>0THEN220
200 D=10
210 C=C-1
220 IFE=65THENE=3.1
230 IFE=66THENE=1.1
240 IFE=67THENE=1.3
250 IFE=68THENE=1.5
260 IFE=69THENE=3.5
270 IFE=70THENE=5.5
280 IFE=71THENE=5.3
290 IFE=72THENE=5.1
300 IFE=74THENE=3.3
310 H=INT(E)
320 K=ABS(H-E)*10
330 GB=(2*A)+(D/5)-(H/30)
340 LB=41+B-(C/8)-(K/48)
370 PRINT"LAT: ";LB,"LON: ";GB
390 RETURN
400 CLS
420 INPUT"VOTRE LOCATOR ";Q$
440 GOSUB70
450 X0=121:Y0=54
460 CURSETX0,Y0,1
470 LA=LB:GA=GB
500 INPUT"LOCATOR DU CORRESPONDANT ";Q$
520 GOSUB70
569 REM+++CALCUL DISTANCE+++
570 DG=GA-GB
580 A=SIN(LA/180*PI)
590 B=SIN(LB/180*PI)
600 C=COS(LA/180*PI)
610 D=COS(LB/180*PI)
620 E=COS(DG/180*PI)
625 X=(A*B)+(C*D*E)
630 ACSX=-ATN(X/SQR(-X*X+1))+1.5708
635 DIST=111.323*(ACSX/PI*180)
640 PRINT"DIST: ";DIST,
650 DC=DIST
669 REM+++CALCUL AZIMUT+++
670 DIST=DIST/1.852
680 R=DIST/60
```

```
690 F=COS(R/180*PI)
700 G=SIN(R/180*PI)
710 X=(B-F*A)/(G*C)
715 ACSX=-ATN(X/SQR(-X*X+1))+1.5708
720 AZI=ACSX/PI*180
730 IFGA-GB>0THENAZI=360-AZI
740 PRINT"AZI: ";AZI;
749 REM+++TRACE DU PARCOURS+++
750 XA=(DC*SIN(AZI/180*PI))/5.6
760 YA=(DC*COS(AZI/180*PI))/5.6
765 IF(X0+XA)>=0AND(X0+XA)<=239AND(Y0-YA)>=0AND(Y0-YA)<=199THEN775
770 XA=0:YA=0:GOTO790
775 PATTERN170
780 DRAWXA,-YA,1
790 CURSETX0,Y0,1
850 IFKEY$=""THEN850ELSECLS
860 DRAWXA,-YA,0:CURSETX0,Y0,1
870 GOTO500
3999 REM+++DESSIN DE LA CARTE+++
4000 HIRES
4010 CURSET115,5,1
4020 FORI=1TO54
4030 READX,Y
4040 DRAWX,Y,1
4050 NEXTI
4900 GOTO400
5000 DATA5,8,21,15,5,-6,4,8,17,2,12,4,17,4,-7,31,-16,18,-3,13,9,-4,
6,8,3,29
5010 DATA8,5,2,8,-19,15,-11,-3,-4,-4,-5,4,-9,-3,-14,10,2,11
5020 DATA-17,5,-15,-8,-17,2,-22,-12,8,-43,8,9,-8,-16,-2,-9,-8,-4,-5
,-14,5,2
5030 DATA-10,-8,-11,-7,-17,-2,-3,-5,5,-2,-4,-2,4,-2,-5,-2,5,-5, 17,
-3
5040 DATA8,4,15,-3,-4,-13,-2,-10,9,0,0,6,22,-2,-3,-5,15,-8,0,-16,10
,-2
5999 REM+++DETERMINE QTH LOCATOR+++
6000 DIMT$(3,3)
6002 TT$="FEDGJCHAB"
6003 K=1
6005 FORI=1TO3
6010 FORJ=1TO3
6015 T$(I,J)=MID$(TT$,K,1)
6016 K=K+1
```

```
6020 NEXTJ
6025 NEXTI
6050 PRINT:PRINT
6055 PRINT"COORDONNEES EN DEGRES DECIMAUX"
6100 INPUT"LAT: ";LA
6105 INPUT"LON: ";GA
6110 PRINT
6119 REM+++LATITUDE+++
6120 B$=CHR$(INT(LA)+25)
6125 M=((LA-INT(LA))*60)/7.5
6150 C=7-INT(M)
6155 C$=STR$(C)
6170 EL=3*(M-INT(M))

6175 IFEL>1THEN6185
6180 EL=1:GOTO6300
6185 IFEL>2THEN6195
6190 EL=2:GOTO6300
6195 EL=3
6299 REM+++LONGITUDE+++
6300 GG=ABS(GA)
6305 G=(INT(GG)*60)+(GG-INT(GG))*60
6310 G=G/120
6320 IFGA<0THEN6335
6325 A$=CHR$(65+INT(G))
6330 GOTO6340
6335 A$=CHR$(90-INT(G))
6340 G=(G-INT(G))*10
6350 IFGA<0THEN6365
6355 D=1+INT(G)
6360 GOTO6370
6365 D=10-INT(G)
6370 IFD<>10THEN6376
6372 D=0
6374 C=C+1:C$=STR$(C)
6376 D$=STR$(D)
6380 EG=(G-INT(G))*3
6390 IFGA<0THEN6440
6400 IFEG>1THEN6420
6410 EG=1:GOTO6500
6420 IFEG>2THEN6435
6425 EG=2:GOTO6500
6435 EG=3:GOTO6500
```

```
6440 IFEG>1THEN6460
6450 EG=3:GOTO6500
6460 IFEG>2THEN6480
6470 EG=2:GOTO6500
6480 EG=1
6500 E$=T$(EL,EG)
6600 Q$=A$+B$+C$+D$+E$
6605 PRINT
6610 PRINTQ$
6615 PRINT
6620 INPUT"AUTRE TRANSFORMATION COORDONNEES->QTH (O/N) ";R$
6630 IFR$="O"THEN6050
6640 GOTO4000
```

CONTEST

Pour les amateurs de concours VHF-UHF, voici un programme qui devrait apporter à l'opérateur, l'aide de l'ordinateur. Ce programme permettra de tenir à jour 500 QSO pour une machine équipée de 48 K RAM.

I) RÔLE DU PROGRAMME

Les différentes tâches confiées au programme sont les suivantes :

1) Tenir à jour de la feuille de concours, pour les possesseurs d'une imprimante.

2) Eviter les QSO en double en testant l'indicatif introduit et en prévenant l'opérateur que la liaison a été établie auparavant.

3) Calculer les distances par le procédé de QTH-Locator, à partir de la position de la station de base.

4) Tenir à jour et présenter à l'opérateur le nombre de points déjà réalisés et la moyenne par QSO.

5) Permettre à tout moment l'interruption du concours, pour les périodes de repos par exemple, sans devoir laisser l'ordinateur sous tension.

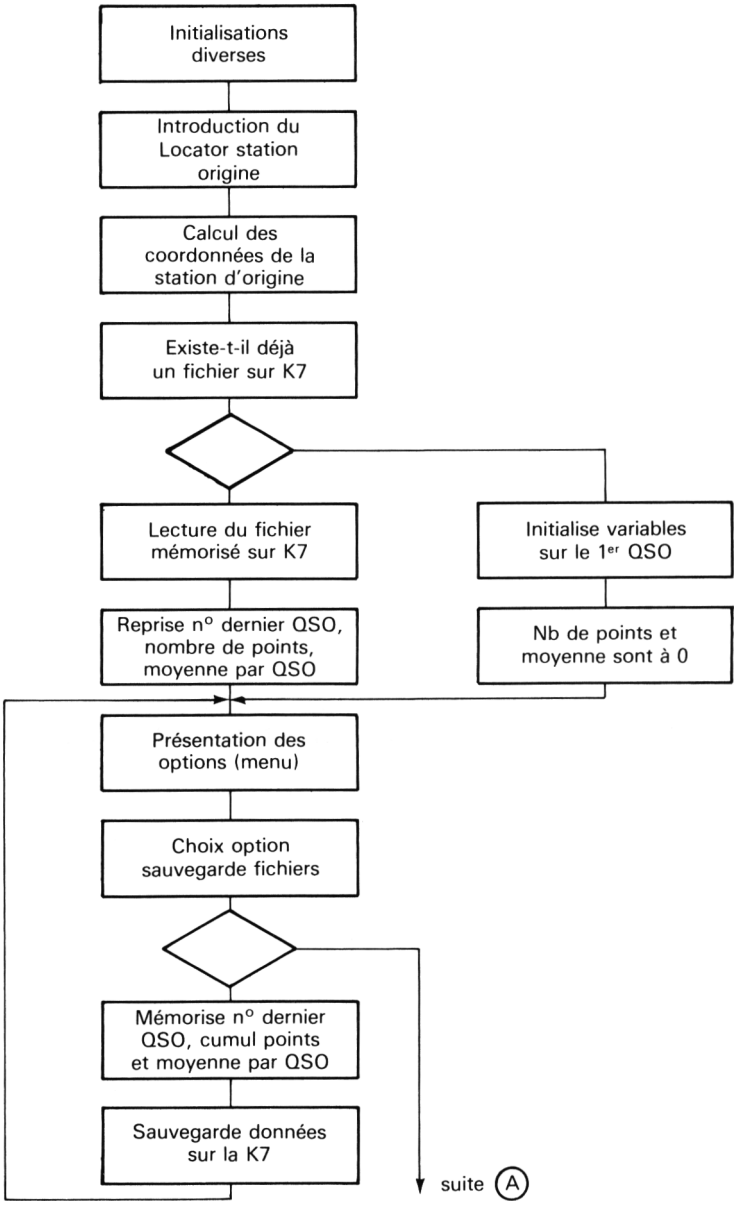
Ce dernier point implique qu'une fonction de SAUVEGARDE et de LECTURE de données soit présente dans le programme.

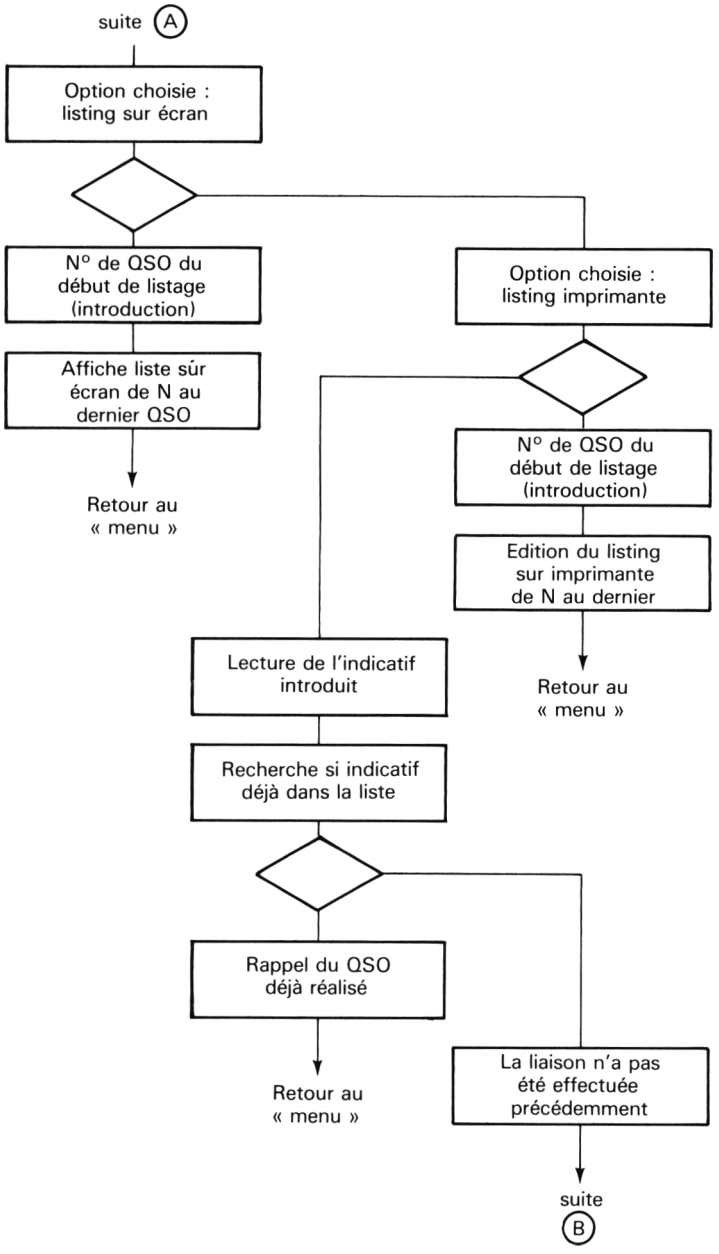
II) SAUVEGARDE DE DONNÉES

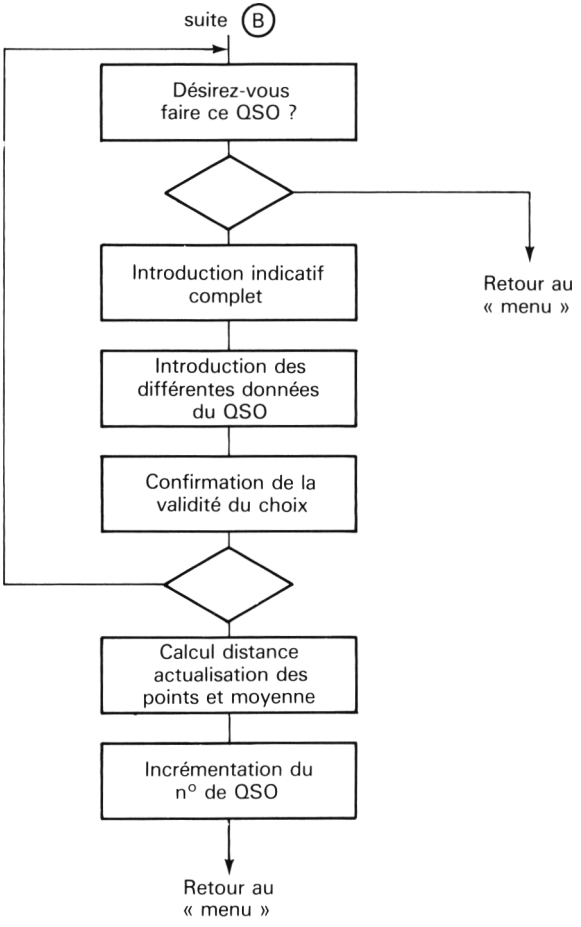
Le BASIC de l'ORIC-1 ne permet pas la sauvegarde de données sur cassettes. Une routine palliant ce défaut avait déjà été publiée. Nous la reprenons ici intégralement et précisons qu'il est très souhaitable que le magnétophone soit équipé d'une télécommande.

Cette routine est implantée à partir de la ligne 20000 du listing BASIC.

III) ORGANIGRAMME SIMPLIFIÉ




suite A
Option choisie : listing sur écran
N° de QSO du début de listage (introduction)
Affiche liste sur écran de N au dernier QSO
Retour au « menu »
Option choisie : listing imprimante
N° de QSO du début de listage (introduction)
Edition du listing sur imprimante de N au dernier
Retour au « menu »
Lecture de l'indicatif introduit
Recherche si indicatif déjà dans la liste
Rappel du QSO déjà réalisé
Retour au « menu »
La liaison n'a pas été effectuée précédemment
suite B



IV) UTILISATION DU PROGRAMME

Nous avons choisi de faire un affichage sur 40 colonnes et en écriture blanche sur fond noir.

Après chargement, le programme attendra votre indicatif et votre QTH Locator.

Si vous aviez déjà commencé le concours et effectué une sauvegarde des QSO réalisés, le programme rechargera le fichier de données. (Attendre le PING en fin de chargement, avant le « menu »).

UN CONSEIL : POUR LE FICHIER DE DONNÉES, UTILISEZ UNE CASSETTE DIFFÉRENTE DE CELLE QUI CONTIENT LE PROGRAMME, POUR EVITER DE DÉTRUIRE CELUI-CI EN CAS DE FAUSSE MANŒUVRE AVEC LE MAGNÉTOPHONE.

Une fois les données réintroduites (temps variable en fonction du nombre de QSO déjà réalisés), le programme affiche un choix d'options. Si vous y répondez en introduisant un indicatif, le programme va rechercher dans sa mémoire (temps variable en fonction des QSO déjà réalisés), si la liaison n'a pas été établie auparavant. Si le QSO a déjà eu lieu, la machine vous rappellera quand... Sinon, vous avez le feu vert pour faire le contact.

La machine vous demande alors l'indicatif complet de la station à contacter. Pourquoi cela ? tout simplement parce que vous êtes autorisé à omettre, lors de la première introduction de l'indicatif pour la recherche, les /P ou /A ou numéro de département. La machine effectuera les tests de comparaison sur l'indicatif seul. Par contre, pour une présentation correcte, avec un maximum de renseignements sur le compte rendu final, vous ajouterez ces éléments manquants.

Le programme attendra ensuite l'introduction des autres données du QSO. Il vous demandera, après la dernière donnée, si vous désirer annuler ou valider ces données (cela, car une erreur est toujours possible lors de l'introduction). Si vous vous apercevez d'une erreur sur les données que vous aurez fournies à la machine, tapez « A » et vous pourrez recommencer.

La machine procède ensuite aux calculs de distance, points cumulés et moyenne par QSO. Noter au passage que nous n'avons pas retenu de notion de multiplicateur. Si ce multiplicateur existe, l'appliquer en fin de compte rendu sur le total général.

Le programme tient constamment à votre disposition les totaux, à chaque affichage du « menu ».

Si vous voulez interrompre le concours, tapez « S » pour sauvegarder les fichiers, et préparez votre cassette en suivant les indications de la machine.

Pour obtenir un listing sur écran, tapez « E » puis le numéro du QSO à partir duquel doit commencer le listing.

Pour obtenir un listing sur imprimante, vérifiez qu'elle est connectée, tapez « P » et le numéro du 1er QSO à lister.

Pour ces deux options, on peut interrompre le listage et revenir au menu en appuyant sur la barre ESPACE.

Au cas, peu probable, où suite à une fausse manipulation, le programme était « planté » faire GOTO 2100 (RETURN) et jamais RUN.

V) COMMENTAIRES SUR LE LISTING

La première partie du listing montre (lignes 70 à 650) le sous-programme de calcul de distance. Nous ne le commenterons pas car il est identique au programme LOCATORIC décrit précédemment.

Notons simplement, à la ligne 630, le calcul de l'arc cosinus, à partir de l'arc tangente. En effet, la fonction arc cosinus n'est pas disponible sur ORIC.

Ligne 1000 : On réserve habituellement une zone mémoire avec l'instruction HIMEM. Ici ce n'est pas le cas. Cette instruction ne sert qu'à certifier la mémoire disponible. En effet, il a été constaté quelques problèmes sur l'ORIC en son absence lors de la manipulation de chaînes nombreuses.

Ligne 1010 : On implante la routine de manipulation de données sur cassette, dans la zone mémoire qui commence en B800 (zone protégée inaccessible au BASIC).

Ligne 1020 : A l'adresse 18 on trouve l'adresse de la position d'affichage. Cette particularité a été utilisée pour écrire, en lieu et place de l'inscription CAPS, le mot « Contest ».

Lignes 1060-1080 : Introduction du locator de la station et détermination des coordonnées pour les calculs de distance. L'indicatif est demandé. Il sera mémorisé et utilisé lors d'un tirage de listing sur imprimante.

Ligne 1150 : Initialisation des tableaux U$ (liste d'indicatifs) et V$ (données associées). Noter que U$ (501 à 503) vont contenir, pour la sauvegarde sur cassette, les valeurs DER (numéro du prochain QSO), CUM (total de points), MOY (moyenne par QSO).

Ligne 1200 : Autorise sur ORIC l'utilisation de 40 colonnes.

Lignes 1220-1440 : Lecture du fichier K7 s'il existe. Noter au passage que l'instruction GET a été préférée à INPUT à chaque fois que possible car elle ne demande pas de RETURN.

Lignes 1400-1410 : Le 0 suivant l'adresse # 67 indique que les fichiers sont enregistrés en vitesse rapide. En vitesse lente, on mettrait un 1. L'adresse 1027 du CALL est celle de la routine de lecture de données.

Lignes 2100-2160 : Présentation des options disponibles et aiguillage en fonction du choix.

Lignes 2170-2300 : Vérifie que l'indicatif introduit n'est pas déjà présent dans la liste, ce qui signifierait que le contact avec la station concernée a déjà eu lieu.

Lignes 2400-2497 : Le programme autorise le contact si vous désirez l'établir. Dans ce cas, il attend l'introduction des données et permet (ligne 2497) leur annulation en cas d'erreur.

Ligne 2505 : Calcul de la distance entre les deux stations.

Lignes 2510-2535 : Imprime les données du QSO.

Lignes 2540-2570 : Réactualise les totaux.

Lignes 3000-3200 : Sous-programme de listage sur écran. Ligne 3020 (équivalent de IF KEY $ ...) on teste s'il y a eu appui sur la barre ESPACE, auquel cas on interrompt le listage.

Lignes 4000-4200 : Sous-programme de listage sur imprimante. Lignes 4010 et 4170 on trouve un DOKE # 306. Ceci compense un défaut de l'ORIC qui, lors de l'utilisation d'une imprimante perd des caractères, à cause du temps de scrutation du clavier. Ce temps de scrutation du clavier est ici modifié pour pallier le défaut. Ligne 4155 CHR $ (14) permet sur la Seikosha GP 100 l'écriture en double largeur. Le retour aux caractères « normaux » est assuré, ligne 4165, par CHR $ (15).

Ligne 5000 : Sous-programme d'appel à la routine de sauvegarde de données sur cassette, avant interruption du CONTEST.
L'adresse 1024 est celle de la routine de transfert des données sur cassette.

Lignes 2000-20180 : Implantation des DATA représentant en hexadécimal, le code machine des routines de manipulation de données sur cassette.

VI) CONCLUSION

En guise de conclusion, nous vous fournissons ici les temps maxima des procédures de recherche de double, sauvegarde de données et rechargement de données.

Ces temps ont été établis avec un fichier de taille maximale, simulé avec 500 QSO.

Noter au passage que le temps de rechargement est un peu plus long que le temps de sauvegarde. Pourquoi ? tout simplement parce que la lecture magnétophone est interrompue par le biais de sa télécommande (il est pratiquement indispensable d'avoir un magnéto à télécommande pour ce programme), pour permettre à l'ordinateur de ranger ses données en mémoire et de re-formater celle-ci.

Recherche : 20 secondes pour 500 QSO
Sauvegarde : 135 secondes pour 500 QSO
Chargement : 175 secondes pour 500 QSO

On est bien loin des temps qui seraient autorisés par un système à disquettes. Remarquons simplement qu'il y a , en moyenne, 35 octets par QSO soit 35 × 8 = 280 bits environ... La vitesse utilisée est de 2 400 bauds...

Il y a quand même 17,5 K octets de mémoire à promener !

* Note pour les possesseurs de magnétophone sans télécommande. Nous suggérons de mettre un PING et un WAIT 1000 en 5405 pour créer un espace entre les 2 fichiers. De même un WAIT 2000 (plus long à la lecture, en 1405) suivi d'un PING pour avoir le temps d'arrêter le magnétophone pendant qu'ORIC organise sa mémoire, pour le redémarrer ensuite.

```
1 REM +++++ CONTEST +++++
2 REM + @  E. DUTERTRE  +
3 REM + @  D. BONOMO    +
4 REM + F1EZH  F6GKQ    +
5 REM +     17-10-1983  +
6 REM +        V.01     +
7 REM +   O R I C - 1   +
8 REM +                 +
9 REM +++++++++++++++++++
10 REM
50 GOTO1000
69 REM---- CALCUL DE DISTANCE ----
70 IFASC(Q$)<84THEN100
80 A=-91+ASC(Q$):GOTO110
100 A=-65+ASC(Q$)
110 Q$=RIGHT$(Q$,4):B=-65+ASC(Q$)
130 Q$=RIGHT$(Q$,3):C=-48+ASC(Q$)
150 Q$=RIGHT$(Q$,2):D=-48+ASC(Q$)
170 Q$=RIGHT$(Q$,1):E=ASC(Q$)
190 IFD<>0THEN220
200 D=10:C=C-1
220 IFE=65THENE=3.1
```

```
230 IFE=66THENE=1.1
240 IFE=67THENE=1.3
250 IFE=68THENE=1.5
260 IFE=69THENE=3.5
270 IFE=70THENE=5.5
280 IFE=71THENE=5.3
290 IFE=72THENE=5.1
300 IFE=74THENE=3.3
310 H=INT(E):K=ABS(H-E)*10
330 GB=(2*A)+(D/5)-(H/30)
340 LB=41+B-(C/8)-(K/48)
390 IFINI=0THENRETURN
570 DG=GA-GB
580 A=SIN(LA/180*PI):B=SIN(LB/180*PI)
600 C=COS(LA/180*PI):D=COS(LB/180*PI)
620 E=COS(DG/180*PI):X=(A*B)+(C*D*E)
630 ACSX=-ATN(X/SQR(-X*X+1))+1.5708
635 DIST=111.323*(ACSX/PI*180)
645 IF(DIST-INT(DIST))>=.5THENDIST=1+INT(DIST)ELSEDIST=INT(DIST)
650 RETURN
999 REM----- INITIALISATIONS -----
1000 HIMEM#97FF
1010 GOSUB20000
1015 INK7:PAPER0
1020 DOKE18,48031:PRINT"Contest"
1050 CLS
1055 PRINT:PRINT:PRINT:PRINT
1060 INPUT"VOTRE LOCATOR ";0$
1062 Q0$=0$
1063 PRINT:PRINT
1065 INPUT"VOTRE INDICATIF ";I$
1070 GOSUB70
1075 LA=LB:GA=GB
1080 INI=1
1150 DIMU$(505),V$(501)
1200 POKE#26A,35
1210 PRINT:PRINT:PRINT
1220 PRINT" AVEZ-VOUS DEJA SAUVEGARDE DES DONNEES ?"
1230 GETRR$
1240 IFRR$<>"O"THENDER=1:GOTO2000
1245 REM----- RELECTURE DU FICHIER ----
```

```
1250 CLS:PRINT:PRINT:PRINT:PRINT
1260 PRINT" REPOSITIONNEZ LA CASSETTE DE DONNEES"
1270 PRINT:PRINT:PRINT" PRESSEZ LA TOUCHE PLAY DU MAGNETO "
1280 PRINT:PRINT:PRINT" ----- PRESSEZ UNE TOUCHE -----"
1290 GETRR$:PING
1300 PRINT:PRINT:PRINT"  PATIENTEZ QUELQUES INSTANTS"
1400 POKE#67,0:CALL1027,U$
1410 POKE#67,0:CALL1027,V$
1420 DER=VAL(MID$(U$(501),2,LEN(U$(501))))
1430 CUM=VAL(MID$(U$(502),2,LEN(U$(502))))
1440 MOY=VAL(MID$(U$(503),2,LEN(U$(503))))
2000 PING:CLS
2100 PRINT:PRINT"TOTAL CUMULE : ";CUM
2102 PRINT"MOYENNE QSO  : ";MOY
2104 PRINT"LE PROCHAIN QSO SERA No : ";DER
2105 PRINT:PRINT
2110 IFDER>500THENPRINT"MAXIMUM DE QSO ATTEINT ":GOTO2125
2120 PRINT:PRINT:PRINT"INTRODUISEZ L'INDICATIF OU:"
2125 PRINT
2130 PRINT"  - S POUR SAUVEGARDER FICHIERS"
2135 PRINT"  - E POUR LISTER SUR ECRAN"
2140 PRINT"  - P POUR LISTER SUR IMPRIMANTE"
2145 PRINT:INPUT"       -> VOTRE CHOIX ";R$
2150 IFR$="S"THEN5000
2155 IFR$="E"THEN3000
2160 IFR$="P"THEN4000
2165 IFDER>500THENCLS:PING:GOTO2110
2170 I=0
2200 REPEAT
2210 I=I+1
2220 IFR$<>MID$(U$(I),1,LEN(R$))THEN2300
2230 CLS
2235 PRINT:PRINT:PRINTR$;" DEJA CONTACTE,QSO NO ";I
2240 PRINT:PRINT:PRINTU$(I);SPC(12-LEN(U$(I)));
2245 PRINTV$(I)
2250 PRINT:PRINT:PRINT:GOTO2100
2300 UNTILI=DER
2400 CLS
2410 PRINT:PRINT:PRINT:PRINT
2415 PRINT"POUR QSO NO ";DER;" VOULEZ-VOUS ";R$;" ?";
2420 GETRR$:PRINTRR$
2425 IFRR$<>"O"THENGOTO2100
2450 PRINT:PRINT:PRINT
```

```
2455 INPUT"INDICATIF COMPLET ";U$(DER)
2460 PRINT
2465 INPUT"QTR DU CONTACT (HHMM) ";QTR$
2470 PRINT
2475 INPUT"QRA LOCATOR ";QRA$
2480 PRINT
2485 INPUT"GROUPE PASSE ";GP$
2490 PRINT
2495 INPUT"GROUPE RECU ";GR$
2496 PING
2497 PRINT:PRINT" A POUR ANNULER V POUR VALIDER ";:GETRR$:IFRR$="A"
THEN2400
2500 PRINT
2505 Q$=QRA$:GOSUB70
2510 CLS:PRINT:PRINT
2515 PRINTU$(DER);:PRINTSPC(12-LEN(U$(DER)));
2530 V$(DER)=QTR$+" "+QRA$+" "+GP$+" "+GR$+" "+STR$(DIST)
2535 PRINTV$(DER):PRINT
2540 CUM=CUM+DIST
2545 MOY=CUM/DER
2570 DER=DER+1
2600 GOTO2100
2999 REM----- LISTAGE SUR ECRAN -----
3000 CLS
3005 PRINT:PRINT:INPUT"LISTAGE A PARTIR DU NO ";N
3010 FORI=NTODER
3020 IFPEEK(#208)=#84THENN=DER:GOTO2120
3100 PRINTU$(I)SPC(12-LEN(U$(I)))V$(I)
3150 NEXT
3200 GOTO2120
3999 REM----- LISTAGE SUR IMPRIMANTE-----
4000 CLS:LPRINT
4005 PRINT:PRINT:INPUT"LISTAGE A PARTIR DU NO ";N
400 DOKE#306,65535
4012 LPRINT:LPRINTCHR$(14);"STATION : ";I$:LPRINT"LOCATOR : ";QQ$
4013 LPRINTCHR$(15):LPRINT
4015 FORI=NTODER
4020 IFPEEK(#208)=#84THENN=DER:GOTO4160
4100 LPRINTU$(I)SPC(12-LEN(U$(I)))V$(I)
4105 PRINTU$(I)SPC(12-LEN(U$(I)))V$(I)
4150 NEXT
4155 LPRINTCHR$(14):LPRINT
4160 LPRINT"No DERNIER QSO : ";DER-1
```

```
4162 LPRINT"TOTAL CUMULE    : ";CUM
4164 LPRINT"MOYENNE PAR QSO: ";MOY
4165 LPRINTCHR$(15)
4170 DOKE#306,10000
4200 GOTO2120
4999 REM---- SAUVEGARDE FICHIERS ----
5000 REM
5020 U$(501)=STR$(DER):U$(502)=STR$(CUM):U$(503)=STR$(MOY)
5250 CLS:PRINT:PRINT:PRINT:PRINT
5260 PRINT" REPOSITIONNEZ LA CASSETTE DE DONNEES"
5270 PRINT:PRINT:PRINT" PRESSEZ LA TOUCHE RECORD DU MAGNETO"
5280 PRINT:PRINT:PRINT" ----- PRESSEZ UNE TOUCHE -----"
5290 GETRR$:PING
5300 PRINT:PRINT:PRINT"  PATIENTEZ QUELQUES INSTANTS"
5400 POKE#67,0:CALL1024,U$
5410 POKE#67,0:CALL1024,V$
5500 PING:CLS:GOTO2120
19999 END
20000 REM------- DONNEES SUR K7 -------
20005 A=#B800:READD$
20010 FORI=1TOLEN(D$)STEP2
20020 V=VAL("#"+MID$(D$,I,2)):POKEA,V:A=A+1:NEXT
20030 READD$:IFD$<>"Z"THEN20010
20040 DOKE#0400,#0A4C:DOKE#0402,#4CB8:DOKE#404,#B858:RETURN
20050 DATA555555552339443638552008B90820D6B820BAE6A92520C6E5A53320C
6E5A53420
20060 DATAC6E520EEB820A7E52428100320358B2004E82860A000B101F017AAA00
2B10199D0
20070 DATA0088D0F8E8CAF008B1D120C6E5C8D0F520C3B890DE602095D5200BB90
820D6B820
20080 DATA96E62030E6C925D0F92030E68533203 0E68534A002B1CEC533C8B1CEE
534B00620
20090 DATA04E84C83C420EEB820EBE4242810032 09BB82004E82860A000B101F01
C20F0D4AA
20100 DATAE8A000CAF0082030E691D1C8D0F5A002B9D00091018 8D0F820C3B890D
96018A903
20110 DATA65018501A89002E602A502C461E5626020CAE62018B9A003B1CEAA88B
1CEE901B0
20120 DATA01CA85338634601 8A5CE65338561A5CF65348562A004B1CE20F6D1855
F84608501
20130 DATA8402602 0E800C92CF0034CE4CF4CE200A20020E80086278 5B420E8002
086D1B006
```

```
20140 DATA2004E84CE4CFA200862886292ØE2009005208 6D1900BAA20E20090FB2
086D1B0F6
20150 DATAC924D006A9FF8528D00CC925D00FA98085290 5B485B48A0980AA20E20
086B5A69E
20160 DATAA59F86CE85CFC5A1D004E4A0F01FA000B1CEC 8C5B4D006A5B5D1CEF00
EC8B1CE18
20170 DATA65CEAAC8B1CE65CF90D738602004E8A22A4C 85C455
20180 DATAZ
```

EXEMPLE D'EXÉCUTION DU PROGRAMME CONTEST
« PRÉSENTATION DES RÉSULTATS »

```
STATION  :  F6GKQ
LOCATOR  :  BI23E


F1EZH/92     1756 BI12E 59001 59001 20
F6ELI/33     1757 ZE19J 59002 57004 477
F1BUU/33     1802 ZE08E 56003 52010 476
EA1CR        1812 XD32A 54004 55012 845


No DERNIER QSO  :  4
TOTAL CUMULE    :  1818
MOYENNE PAR QSO :  454.5
```

PROGRAMME MONIMORSE

Le programme que nous présentons maintenant est entièrement écrit en BASIC. Pourquoi ce choix ? Simplement pour proposer une introduction qui vous aidera à mieux saisir certains principes du programme en langage machine qui suit.

Ce premier programme ne permet pas le décodage. Il permet en fait d'utiliser les fonctions sonores de l'ORIC pour générer les signaux MORSE. Il pourrait donc être aisément modifié pour une transposition sur un autre micro-ordinateur doté d'un générateur sonore.

Vous remarquerez, par ailleurs, que nous avons représenté les données à émettre, directement par leur CODE MORSE. C'est l'objet des lignes de DATA situées en fin de programme.

L'organigramme simplifié et les explications suivantes permettront de bien comprendre ce programme.

BUT DU PROGRAMME

Le but visé est l'enseignement de la CW par un moniteur inlassable. Ce professeur vous offrira plusieurs possibilités.

1. Une dictée avec seulement des lettres. Ceci est une manière de commencer l'apprentissage.
2. Une dictée avec chiffres, lettres et signes utilisés fréquemment en CW, pour parfaire la leçon.
3. Une option de tout début où, lorsqu'on appuie sur une touche du clavier, le code correspondant est émis et le profil point-trait affiché à l'écran en même temps que le caractère.
4. Une option « texte » où l'on peut composer un texte complet pour le générer ensuite. Ceci a l'avantage sur la dictée de pouvoir familiariser l'opérateur avec des textes en clair et les diverses procédures de trafic en CW.

Nous pensons que l'éventail des possibilités ainsi proposées est complet et que chacun y trouvera son niveau.

Complétons cette présentation en soulignant que, lors des dictées, les groupes de signes sont émis par 5. La longueur totale de la dictée, comme d'ailleurs celle du texte préparé, ne peut excéder 255 caractères.

Précisons, enfin, que la vitesse « d'émission » est réglable ainsi que le volume de son produit, par le programme.

PRINCIPE RETENU

IL est fort simple. Il s'agit de « convertir » le code d'un caractère pris au clavier ou lu dans une chaîne, en code MORSE, c'est-à-dire en une succession de points et de traits.

Pour ce faire, on a recours à une table de transcodage qui établira la relation entre le caractère et son code MORSE correspondant. Nous avons donc rangé dans cette table tous les codes MORSE en regard des caractères correspondants. La liaison entre les deux est établie en fonction du code ASCII du caractère.

Ainsi le dernier caractère de la table est rangé à l'emplacement 59 et c'est la lettre Z. Le Y est en 58, le X en 57, etc...

Pour transcoder, nous aurions pu utiliser des profils binaires, successions de 1 et de 0 pour représenter les points et les traits. Ceci se justifie pleinement, nous le verrons, dans le cas d'un programme écrit en langage machine. Ici, nous avons préféré utiliser un codage plus évident, permis par le BASIC et nous avons choisi le code... MORSE, tout simplement.

C'est ce code que l'on trouve dans les DATA des dernières lignes. Certains caractères n'y sont pas représentés et son remplacés par le signe équivalent. Vous trouverez ainsi la virgule, le tiret, les deux-points, le point, le point-virgule. Si vous désirez disposer de ces caractères lors des dictées il suffira de modifier l'emplacement correspondant dans les lignes de DATA ainsi que le OR X = dans la ligne 245.

EXAMEN DU LISTING DE MONIMORSE

Lignes 100-130	: Remplit la table avec le code MORSE rangé dans les lignes de DATA.
Ligne 140	: Crée et initialise la table qui sera remplie lors de la composition de la dictée.
Ligne 145	: PRINT CHR $ (6) rend le clavier muet (inhibe le bip sonore des touches).
Lignes 150-180	: Propose le menu.
Ligne 185	: Selon qu'on choisit ou non le mode clavier, on fera patienter l'utilisateur, car la composition de la dictée demande du temps.
Ligne 190	: L'ordre ON GOTO permet l'aiguillage vers les différentes parties du programme, selon l'opinion choisie.
Lignes 200-210	: Définition des 2 fonctions qui permettent le tirage aléatoire parmi les lettres seules (à partir du code 65) ou en incluant les signes (à partir du code 43).
Lignes 232-265	: Remplissage de la table DI$ qui contient la dictée, par paquets de 5 caractères espacés d'un blanc. La ligne 245 permet un nouveau tirage aléatoire si le caractère

sorti ne possède pas d'équivalent en MORSE ou si on ne veut pas l'entendre (ponctuation).

Ligne 285 : TE$ est la chaîne qui contiendra toute la dictée.

Ligne 290 : On transfère dans TE$ toute la table DI$.

Ligne 295 : Définition de la vitesse et, en fonction, calcul de E qui représente la valeur du temps élémentaire.

Ligne 296 : Le choix du volume a été rendu programmable.

Ligne 297 : Si on a choisi l'option clavier, on affiche à chaque fois le mode d'emploi. Le signe B sert d'indicateur de fin d'utilisation de l'option.

Lignes 300-360 : Saisie du caractère à coder et génération du son. On utilise les fonctions de découpage de chaîne pour isoler le caractère qui nous intéresse.

Ligne 304 : Si le caractère est un espace, on attend 5 fois la durée du temps élémentaire, définie par E.
Pour les utilisateurs d'un autre type de machine, signalons que WAIT N provoque sur ORIC une Pause dans le programme de n fois 10 ms.

Ligne 310 : En mode clavier, on affichera le caractère. En mode dictée, non.

Lignes 325-340 : PLAY 1, 0, 0, 0 ouvre le générateur sonore et MUSIC 1, 4, 10, VOL génère le son. Les possesseurs d'autres micro-ordinateurs utiliseront les fonctions équivalentes. La durée du son est D, celle du silence, E.

Ligne 345 : On attend 1 fois E entre 2 « bits » d'un même caractère.

Ligne 355 : On attend 3 fois E entre 2 caractères différents.

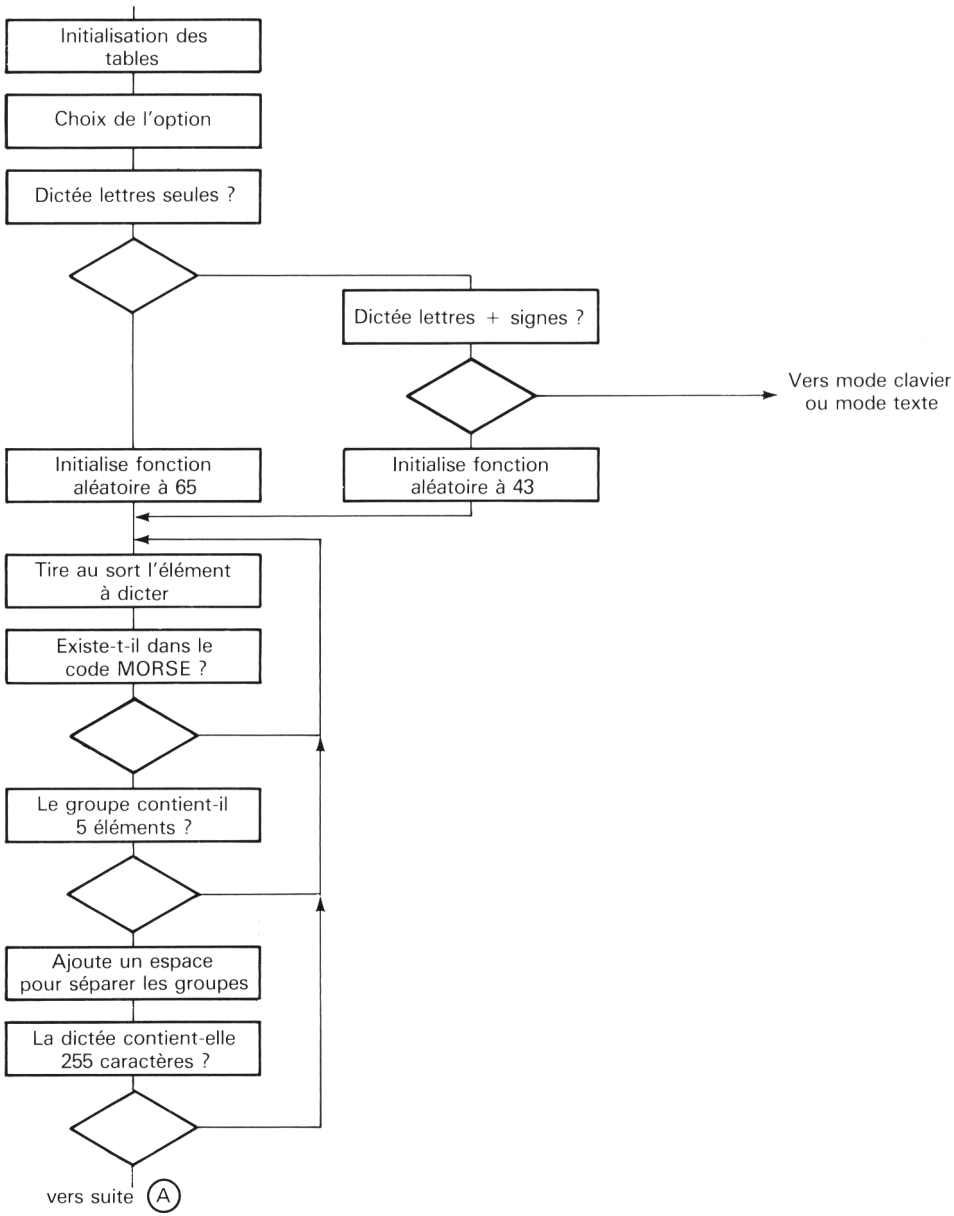
Lignes 380-400 : On propose, en fin de dictée, un menu différent qui permet de changer d'option ou d'obtenir le corrigé de la dictée précédente.

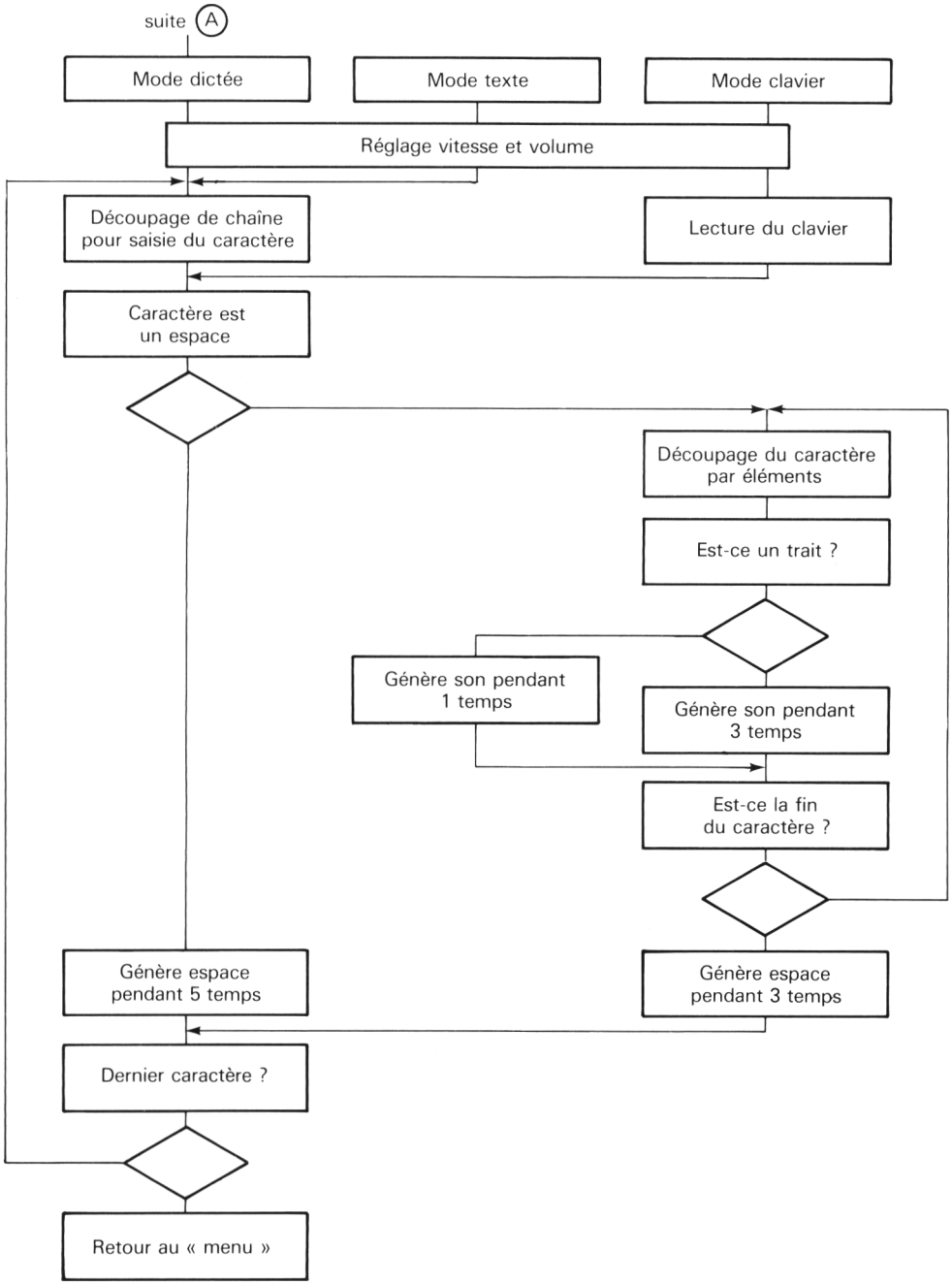
Lignes 600-730 : Cette partie du programme autorise l'introduction d'un texte, de 255 caractères (ou espaces) maximum, qui pourra être émis. C'est l'option 4.

Lignes 1025-1070 : On trouve les lignes de DATA qui contiennent le profil MORSE des caractères. Les caractères non utilisés sont remplacés par leur symbole (ex. : virgule, tiret, etc...), ce qui permettra une modification aisée du programme pour qui voudra générer ces caractères. Ne pas omettre alors la modification de la ligne 245 en supprimant le OR X = correspondant (le code suivant le OR X est le code ASCII du caractère).
44 : la virgule
45 : le tiret
46 : le point
58 : les deux-points
59 : le point-virgule
60, 62, 64 non utilisés en MORSE.

Pour conclure cette description, signalons que nous avons retenu les critères d'espace suivants, en considérant que la durée du point est E, la durée du trait 3 fois E.

E : espace entre 2 bits d'un même caractère
3 fois E : espace entre 2 caractères
5 fois E : durée du caractère « espace » (espace entre 2 mots)
Ces critères peuvent, bien entendu être modifiés.


Initialisation des tables
Choix de l'option
Dictée lettres seules ?
Dictée lettres + signes ?
Vers mode clavier ou mode texte
Initialise fonction aléatoire à 65
Initialise fonction aléatoire à 43
Tire au sort l'élément à dicter
Existe-t-il dans le code MORSE ?
Le groupe contient-il 5 éléments ?
Ajoute un espace pour séparer les groupes
La dictée contient-elle 255 caractères ?
vers suite A


suite A
Mode dictée
Mode texte
Mode clavier
Réglage vitesse et volume
Découpage de chaîne pour saisie du caractère
Lecture du clavier
Caractère est un espace
Découpage du caractère par éléments
Est-ce un trait ?
Génère son pendant 1 temps
Génère son pendant 3 temps
Est-ce la fin du caractère ?
Génère espace pendant 5 temps
Génère espace pendant 3 temps
Dernier caractère ?
Retour au « menu »

```
10 REM    +++++ MONIMORSE +++++
15 REM    +  @ EDDY  DUTERTRE +
20 REM    +  @ DENIS BONOMO   +
25 REM    +  F1EZH   F6GKQ    +
30 REM    +     04-08-1983    +
35 REM    +       (V.01)      +
40 REM    +    O R I C - 1    +
45 REM    +                   +
50 REM    +++++++++++++++++++++
55 REM
99 REM+++ INITIALISATIONS +++
100 DIMTA$(59)
110 FORI=12TO59
120 READTA$(I)
130 NEXT
140 DIMDI$(255)
145 PRINTCHR$(6)
149 REM+++ MENU +++
150 CLS
155 PRINT:PRINT
160 PRINT"    1 POUR DICTEE LETTRES SEULES"
165 PRINT"    2 POUR DICTEE COMPLETE "
170 PRINT"    3 POUR CLAVIER "
171 PRINT"    4 POUR COMPOSER UN TEXTE"
172 PRINT
175 PRINT"                    VOTRE CHOIX ";
180 INPUTC
182 PRINT
185 IFC< 3THENPRINT"PATIENTEZ UN INSTANT"ELSEPRINT
190 ONCGOTO200,210,295,600
199 REM+++ COMPOSE DICTEE +++
200 DEFFNA(X)=INT(RND(1)*26+65)
205 GOTO230
210 DEFFNA(X)=INT(RND(1)*48+43)
230 K=0
232 FORI=1TO5
235 X=FNA(X)
240 IFC=1THEN255
245 IFX=44ORX=45ORX=46ORX=58ORX=59ORX=60ORX=62ORX=64THEN235
255 DI$(K)=CHR$(X)
260 K=K+1
```

```
265 NEXTI
270 DI$(K)=" "
275 K=K+1
280 IFK<250THEN232
285 TE$=""
289 REM+++ TEXTE PREPARE +++
290 FORI=0TO255:TE$=TE$+DI$(I):NEXT
291 PLOT0,7,"                              "
295 INPUT"VITESSE (1 A 10) ";V:E=20/V
296 INPUT"VOLUME (1 A 15) ";VOL
297 IFC<>3THEN300ELSEPRINT"PRESSER LA TOUCHE A CODER OU @ "
298 GETTE$:IFTE$<>"@"THEN300ELSE150
299 REM+++ EMISSION +++
300 FORJ=1TOLEN(TE$):C$=MID$(TE$,J,1)
302 IFC$<>" "THEN305
303 IFC<>3THEN304ELSEPRINTC$
304 WAIT5*E:GOTO360
305 IFASC(C$)>900RASC(C$)<43THENR=1 ELSER=ASC(C$)-31
310 IFC<>3THEN315
312 PRINTC$,TA$(R)
315 FORI=1TOLEN(TA$(R))
320 IFMID$(TA$(R),I,1)="~"THEND=(3*E)ELSED=E
325 PLAY1,0,0,0
330 MUSIC1,4,10,VOL
335 WAITD
340 PLAY0,0,0,0
345 WAITE
350 NEXTI
355 WAIT3*E
360 NEXTJ
365 IFC=3THEN297
370 CLS
375 PRINT:PRINT
379 REM+++ NOUVELLE OPTION +++
380 PRINT"    1 POUR EMISSION DU MEME TEXTE"
385 PRINT"    2 POUR UNE AUTRE OPTION"
387 PRINT"    3 POUR OBTENIR LE CORRIGE"
390 PRINT
395 INPUT"          VOTRE CHOIX ";CC
397 PRINT
400 ONCCGOTO295,150,500
499 REM+++ CORRIGE DICTEE +++
500 CLS
```

```
505 PRINT:PRINT
510 PRINT"------ VOICI LE TEXTE EMIS ------"
515 PRINT
520 PRINTTE$
530 PRINT
540 PRINT
550 PRINT"APPUYER SUR UNE TOUCHE POUR CONTINUER"
560 GETT$:GOTO150
599 REM+++ TEXTE COMPOSE CLAVIER +++
600 CLS
605 PRINT:PRINT
610 PRINT"MAXIMUM 255 CARACTERES. @ POUR FINIR"
620 PRINT"--------------------------------------------"
630 PRINT:PRINT
635 TE$=""
640 I=0
660 GETC$
665 I=I+1
670 IFI=256ORC$="@"THEN710
680 TE$=TE$+C$
690 PRINTC$;
700 GOTO660
710 PRINT
720 PRINT
730 GOTO295
999 REM+++ DONNEES TRANSCODAGE +++
1025 DATA".-.-.","","","-","."
1030 DATA"-----",".----","..---","...--","....-","....."
-...."
1035 DATA"---..","----."
1040 DATA":",")","<","-...-",">","..--.."," "
1050 DATA".-","-...","-.-.","-..",".","..-.","--.","....","..",".-
-","-.-"
1060 DATA".-..","--","-.","---",".--.","--.-",".-.","...","-",".-
,"...-"
1070 DATA".--","-..-","-.--","--.."
```

MANIPORIC

MANIPORIC est un programme tout simple que nous avons choisi pour illustrer le passage du BASIC au langage machine. Nous transformons en fait la machine en un manipulateur MORSE qui génèrera des points à chaque appui sur une touche, des traits en actionnant une autre touche. La vitesse sera rendue variable, mais l'écart entre les limites de cette variation sera faible, car il n'est pas facile d'utiliser les touches de l'ORIC comme manipulateur.

Cette vitesse sera évaluée en nombre de mots (ceci est assez approximatif). Nous avons considéré pour calculer la durée du point élémentaire, des mots de 5 lettres. On peut ainsi calculer le nombre de points par minute, tablant sur la relation suivante :

3 points pour la durée d'un trait.
1 point pour l'espace entre « bits » d'un même caractère.
3 points pour l'espace entre 2 caractères.
5 points pour l'espace entre 2 mots.

Si l'on prend le mot TRAFIC, on a :

T = —	3 + 3 =	6 points
R = . — .	=	10
A = . —	=	8
F = . . — .	=	12
I = . .	=	5
C = — . — .	=	14
Séparation mot :	=	5
		60 points

ce qui donnerait, à 10 mots/mn, environ 600 points/mn.

Les durées de point correspondantes sont donc :

10 mots : 100 ms
15 mots : 66 ms
20 mots : 50 ms
30 mots : 33 ms etc...

Les touches choisies sont Z pour les points et V pour les traits (correspondent à l'index et au pouce de l'opérateur gaucher !). Toute action sur une autre touche permettra de sortir du programme.

Le programme BASIC est le reflet du petit organigramme de principe poposé. Il utilise le générateur sonore. On générera un son court pour appui sur Z et long pour appui sur V.

Les variables P et T représentent les durées, multiples de 10 ms (emploi de l'instruction WAIT), du Point et du Trait.

PRINT CHR$ (6) permet d'inhiber le bip sonore sur les touches. La bascule est repositionnée pour le réautoriser en quittant le programme, par la même instruction.

POKE # 307, 29 accélère le cycle de scrutation du clavier, ce qui a pour effet de raccourcir le temps entre deux saisies.

Notons que, grâce à la fonction GET, le traitement ne commence que lors de l'appui sur une touche.

MANIPORIC VERSION BASIC

A titre d'exemple nous vous proposons maintenant la version de ce même programme mais en langage machine. Cela permettra de revoir, entre autre, la programmation du circuit sonore, expliquée précédemment.

Nous avons conservé les initialisations en BASIC et le nombre minimum de mots par minute a changé, pour simplifier l'initialisation des variables qui contiendront les durées du Point et du Trait (adresses 0400 et 0401).

L'action sur la touche Z permet l'émission des points ; toute action sur une autre touche provoquera l'émission des traits. Pour sortir du programme, actionner la barre ESPACE.

La routine en langage machine est située en # 9700.

Sa structure est :

— programmation du circuit sonore
— s'assure qu'aucune touche n'est actionnée

Boucle principale :

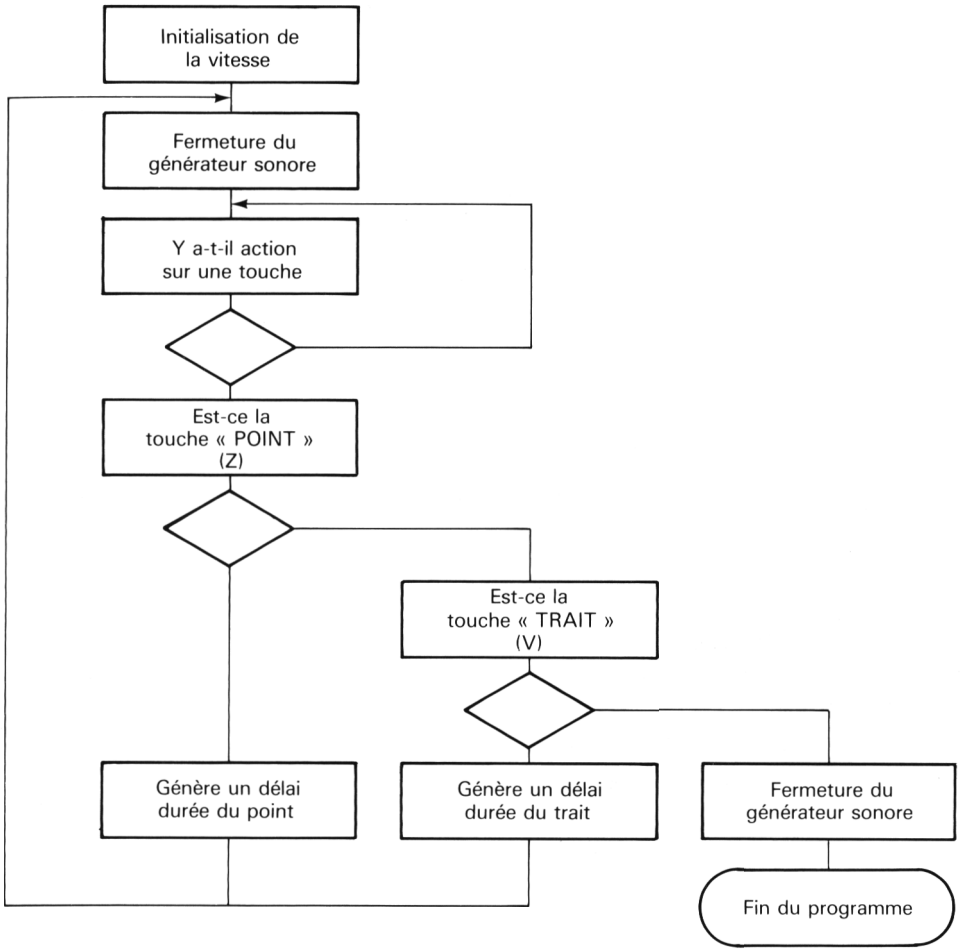
— saisie du clavier (par routine en ROM # C5F8)
— test de la touche actionnée
— ouverture du générateur sonore pendant le temps du POINT ou du TRAIT
— fermeture du générateur sonore
— délai durée un POINT assurant l'espace entre bits d'un même caractère

Pour changer la fréquence de la note émise, il suffit de modifier la valeur (# 64) qui se trouve en # 9710 par

POKE # 9710, (valeur désirée)

Pour calculer la valeur désirée, reportez-vous au chapitre sur le AY-8912.

Le volume est en # 9712.



ORGANIGRAMME DE MANIPORIC

```
1 REM ******* MANIPORIC *******
2 REM *     @ Eddy DUTERTRE     *
3 REM *     @ Denis BONOMO      *
4 REM *          F1EZH          *
5 REM *          F6GKQ          *
6 REM *                         *
7 REM *     19-11-1983 (V01)    *
8 REM *       O R I C - 1       *
9 REM *                         *
10 REM***************************
15 REM
20 CLS:PAPER0:INK2
25 POKE#307,29
30 PRINT:PRINT:PRINT
40 INPUT"Nombre de mots/mn (5 a 30) ";N
45 PRINT:PRINT
47 PRINT"   Z  Pour les POINTS                          V  Pour les TRAIT
S "
50 PRINTCHR$(6)
60 PRINT:PRINT"Toute autre touche Pour arreter"
70 P=100/N:T=300/N
100 PLAY0,0,0,0:GETA$
150 PLAY1,0,0,0:MUSIC1,5,5,8
200 IFA$="Z"THENWAITP:GOTO100
300 IFA$="V"THENWAITT:GOTO100
400 PLAY0,0,0,0:PRINTCHR$(6):END
```

MANIPORIC VERSION LANGAGE MACHINE

```
1 REM------ MANIPORIC ------
2 REM
100 HIMEM#9700
110 FORI=#9700TO#976A:READA$
120 IN=VAL("#"+A$):POKEI,IN:NEXT
130 CLS:PAPER0:INK2
150 PRINT:PRINT:PRINT
200 INPUT"Nombre de mots/mn (>= a 12) ";N
205 POKE#307,29
210 POKE#400,1E3/N:POKE#401,3E3/N
220 PRINT:PRINT:PRINT"  Z Pour les Points"
225 PRINT:PRINT"  V Pour les traits"
```

```
230 PRINT:PRINT"  ESPACE Pour quitter"
250 PRINTCHR$(6):CALL#9700:PRINTCHR$(6)
260 END
300 DATAA9,01,A2,00,20,35,F5,A9,00,A2,64,20,35,F5,A9,08,A2,08,20,35
,F5
310 DATAA9,07,A2,FF,20,35,F5,AD,08,02,C9,38,D0,F9
320 DATA20,F8,C5,C9,20,D0,01,60,8D,02,04
330 DATAA9,07,A2,FE,20,35,F5,EA,EA,EA,EA,EA
340 DATAAD,02,04,C9,5A,D0,0C,AC,00,04,20,5F,97
350 DATA20,53,97,4C,23,97,AC,01,04,4C,44,97
360 DATAA9,07,A2,FF,20,35,F5,AC,00,04,EA,EA
370 DATAA2,C7,CA,D0,FD,88,D0,F8,AC,00,04,60
```

ASSEMBLEUR

```
0B
9700 A901   LDA %#01
9702 A200   LDX %#00
9704 2035F5 JSR #F535
9707 A900   LDA %#00
9709 A264   LDX %#64
970B 2035F5 JSR #F535
970E A908   LDA %#08
9710 A208   LDX %#08
9712 2035F5 JSR #F535
9715 A907   LDA %#07
9717 A2FF   LDX %#FF
9719 2035F5 JSR #F535
971C AD0802 LDA #0208
971F C938   CMP %#38
9721 D0F9   BNE #971C
9723 20F8C5 JSR #C5F8
9726 C920   CMP %#20
9728 D001   BNE #972B
972A 60     RTS
972B 8D0204 STA #0402
972E A907   LDA %#07
```

Programme réglage « GROS » de la note

Programme réglage « FIN » de la note

Programme réglage du VOLUME

Ferme le générateur sonore (précaution)

S'assure qu'aucune touche n'est actionnée (précaution)

Saisie du clavier par routine en ROM

Retour au BASIC si ESPACE
Mémorise touche actionnée

Ouvre le générateur sonore

```
9730 A2FE    LDX %#FE
9732 2035F5  JSR #F535
9735 EA      NOP
9736 EA      NOP
9737 EA      NOP
9738 EA      NOP
9739 EA      NOP
973A AD0204  LDA #0402
973D C95A    CMP %#5A
973F D00C    BNE #974D
9741 AC0004  LDY #0400
9744 205F97  JSR #975F
9747 205397  JSR #9753
974A 4C2397  JMP #9723
974D AC0104  LDY #0401
9750 4C4497  JMP #9744
9753 A907    LDA %#07
9755 A2FF    LDX %#FF
9757 2035F5  JSR #F535
975A AC0004  LDY #0400
975D EA      NOP
975E EA      NOP
975F A2C7    LDX %#C7
9761 CA      DEX
9762 D0FD    BNE #9761
9764 88      DEY
9765 D0F8    BNE #975F
9767 AC0004  LDY #0400
976A 60      RTS
```

Récupère le code de la touche actionnée
Teste si c'est « Z » (= Point)

Charge avec valeur durée du point et délai

Ferme générateur

Cas du « TRAIT »

Sous-programme de fermeture du générateur sonore puis délai d'un point

Sous-programme de délai

LE MORSE SUR ORIC-1

Ce programme appelé ORICMORSE sera utile nous le pensons aussi bien au manipulateur chevronné qu'au débutant. En effet, une fois entré dans votre ORIC, il vous donnera le choix entre trois possibilités que nous allons essayer de décrire.

1) OPTION EMISSION

Après avoir choisi la vitesse à laquelle vous voudrez émettre votre message (de 0 à 10) on vous demandera si vous désirez également la commutation. En effet, il a été prévu de pouvoir commander directement votre émetteur par le relais interne de l'ORIC. Dans le cas où vous ne le souhaitez pas, répondez Non pour lui éviter de travailler inutilement. Ensuite, toute touche appuyée sera immédiatement transmise en morse par le générateur sonore interne et, si l'option a été choisie, également par le relais télécommande du magnétophone. Il faut également préciser que le volume sonore est réglable par les deux touches curseur « ↑↓ ». Il est également possible d'émettre un message mémorisé en appuyant sur la touche #. Ce message ne peut avoir au maximum que 254 caractères et doit être rentré au préalable dans les lignes 505-510 et 515 du Basic (variable A$).

Pour sortir de l'option émission, il vous suffira alors d'appuyer sur la touche « Return ».

2) OPTION RÉCEPTION

Elle vous permettra de décoder des messages morse en connectant l'interface dont vous trouverez le schéma ci-après. Cette interface permet donc la transformation des signaux BF issus d'un récepteur décamétrique, par exemple, en signaux logiques immédiatement exploitables par l'ORIC. Ces signaux seront collectés par la prise imprimante et plus précisément par la ligne Acknoledge. Pour ce qui est du réglage de la vitesse, il n'y a pas de problème puisque l'ordinateur s'asservit automatiquement dans la mesure où bien sûr la manipulation décodée est correcte. En cours de décodage, dès que l'écran est plein, il y a effacement automatique, le scrolling prenant trop de temps.

Pour sortir de cette option, tout comme l'émission, il suffit d'appuyer sur « return ».

3) OPTION ENTRAÎNEMENT

Dans cette option, l'ordinateur va créer une dictée aléatoire qu'il vous soumettra à la vitesse de votre choix. A la fin de la dictée, le corrigé sera affiché à l'écran et il vous sera alors possible de recommencer.

Voila donc décrit le fonctionnement ; il ne vous reste plus qu'à faire avaler le programme à l'ordinateur. Attention aux lignes DATA qui sont très importantes puisqu'elles contiennent le langage machine.

DESCRIPTION DU PROGRAMME BASIC

Lignes 20 à 87	: Changement langage machine contenu dans les lignes data 1000 à 1590
Lignes 90 à 160	: Présentation et choix de l'option
Lignes 400 à 500	: Préparation et appel routine décodage (# 8009)
Lignes 505 à 430	: Préparation et appel routine émission (# 820D)
Lignes 800 à 1000	: Sous-programme entraînement

SYNOPTIQUE EMISSION CARACTÈRE

Pour chaque caractère, un trait est symbolisé par un 1 et point par un 0. De plus, comme chaque code morse a un nombre de points et de traits variable, un bit de start est nécessaire.

EXEMPLE :

F = . . — . = 00010010
↓

bit start

La table de transcodage ASCII → MORSE commence à l'adresse # 8600

SYNOPTIQUE ROUTINE DÉCODAGE

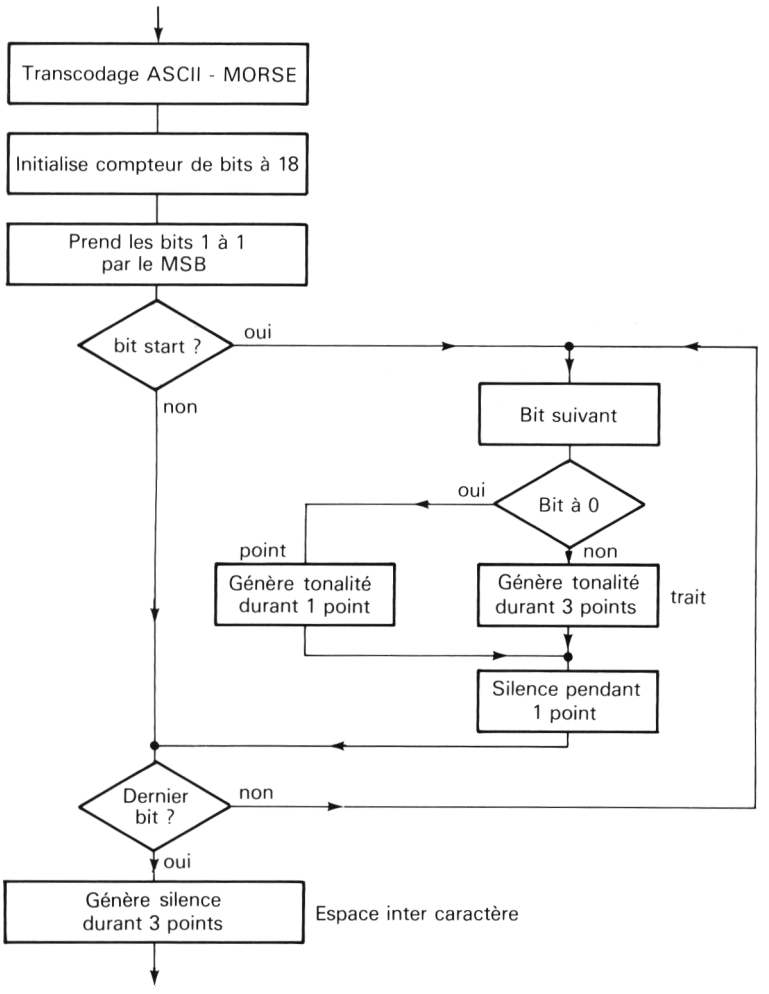
Cette routine fonctionne par la mesure de la durée du signal logique entrant. Si ce signal est à 1, il s'agit donc d'un élément (point ou trait). Pour déterminer sa nature et s'affranchir des variations de vitesse, la durée de l'élément mesuré est comparée à celle du dernier point reçu. Sa durée est ensuite mémorisée et suivant le cas on ajoute 1 à l'une des variables comptabilisant les points et les traits.

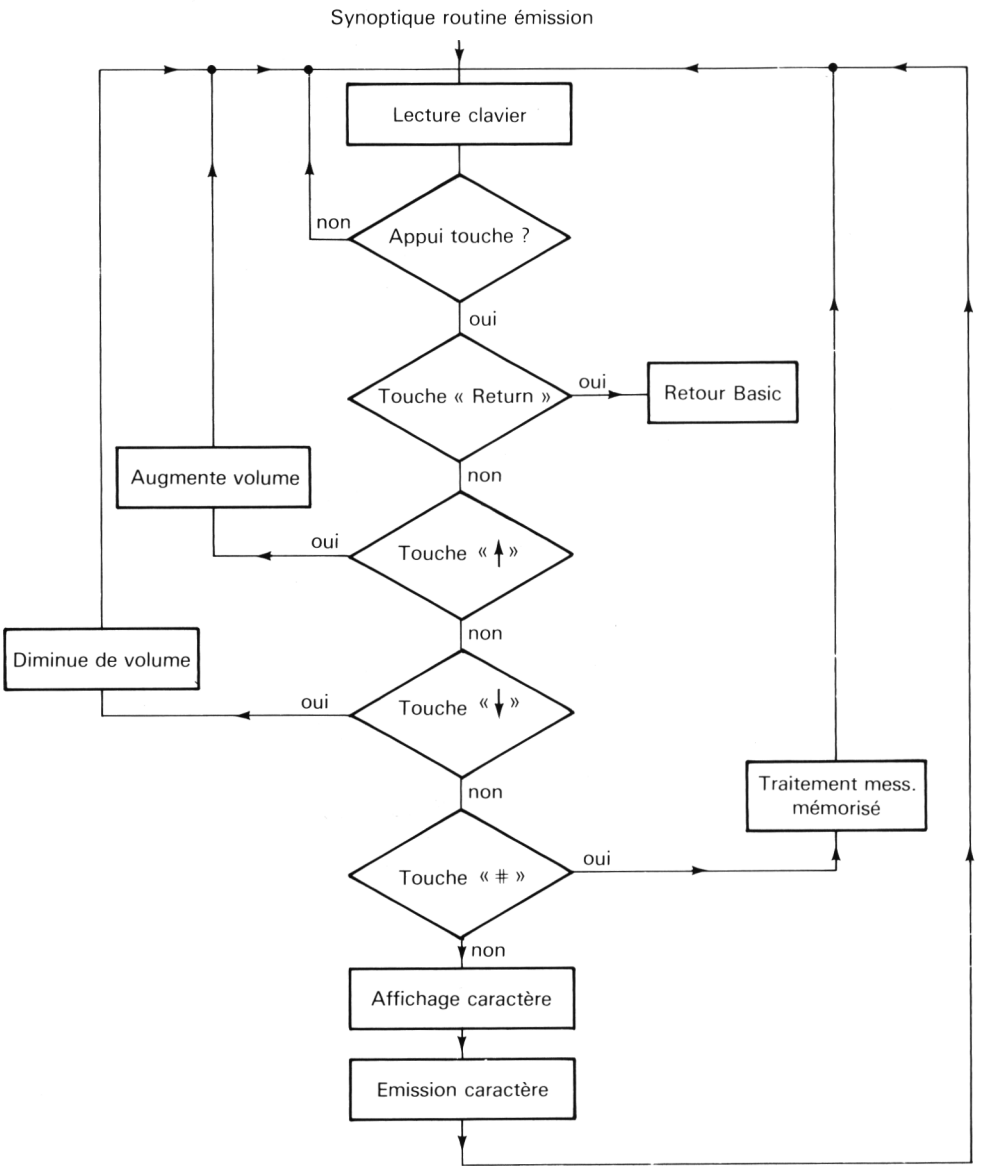
L'affichage du caractère se fait dès la réception d'un espace (niveau logique 0) de durée supérieure ou égale à 3 points.

DÉTERMINATION POINT OU TRAIT

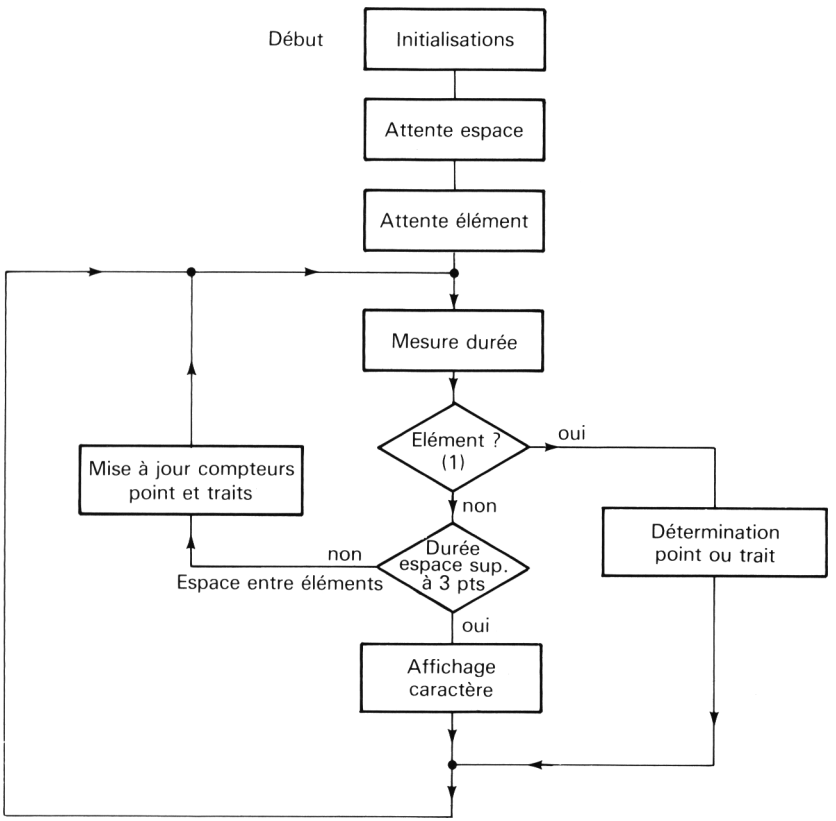
DR = durée signal reçu
DP = durée signal précédent

Synoptique émission caractères



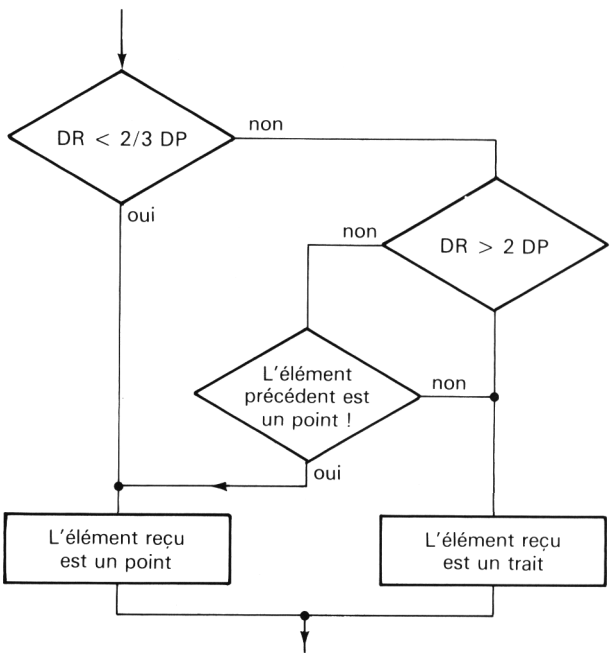

Synoptique routine émission
Lecture clavier
non
Appui touche ?
oui
Touche « Return »
oui
Retour Basic
non
Augmente volume
oui
Touche « ↑ »
non
Diminue de volume
oui
Touche « ↓ »
non
Traitement mess. mémorisé
Touche « # »
oui
non
Affichage caractère
Emission caractère

Synoptique routine décodage



Détermination point ou trait

DR = durée signal reçu
DP = durée signal précédent

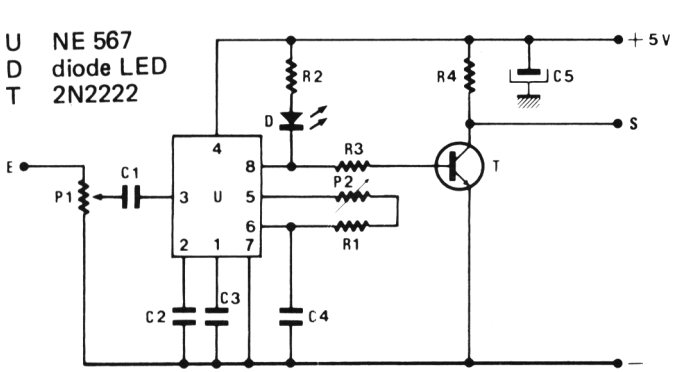


Rappel des variables : DR = durée signal reçu
DP = durée signal précédent
L = lecture précédente du VIA
T = type du signal précédent (0 = point – 1 = trait)
P = durée dernier point reçu

SCHÉMA INTERFACE ET RACCORDEMENT A L'ORIC

P_2 sert à régler la plage de capture de la P.L.L.
Le changement de la diode LED doit être franc au rythme de la BF.
Le récepteur doit être muni d'un bon filtre étroit...

CIRCUIT INTERFACE

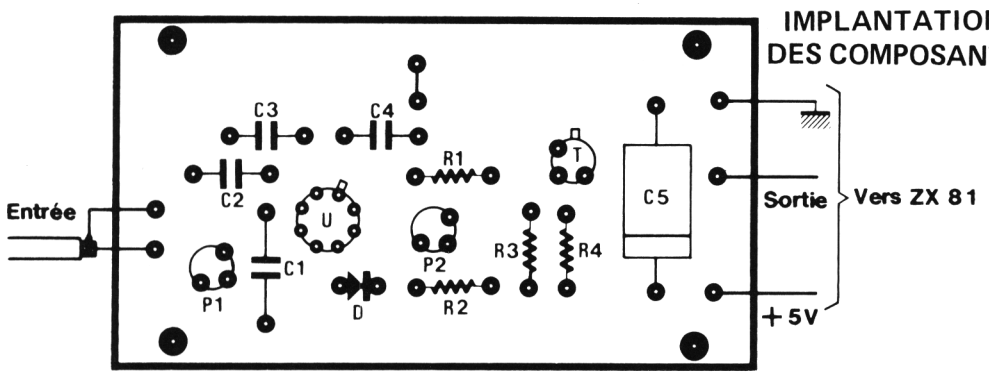


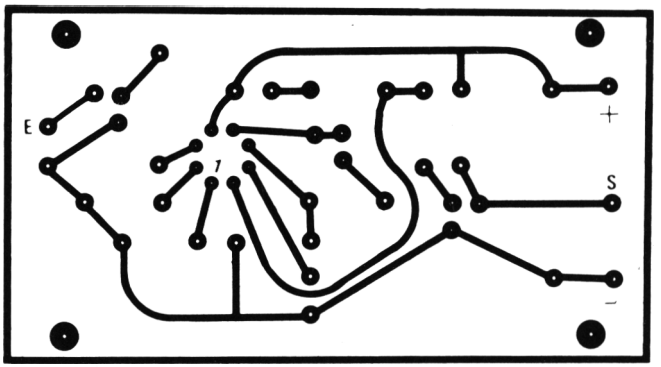
R1 8,2 kΩ
R2 1 kΩ
R3 10 kΩ
R4 10 kΩ

P1 potent. 1 kΩ
P2 potent. 10 kΩ multi tours
C1 0,22μ

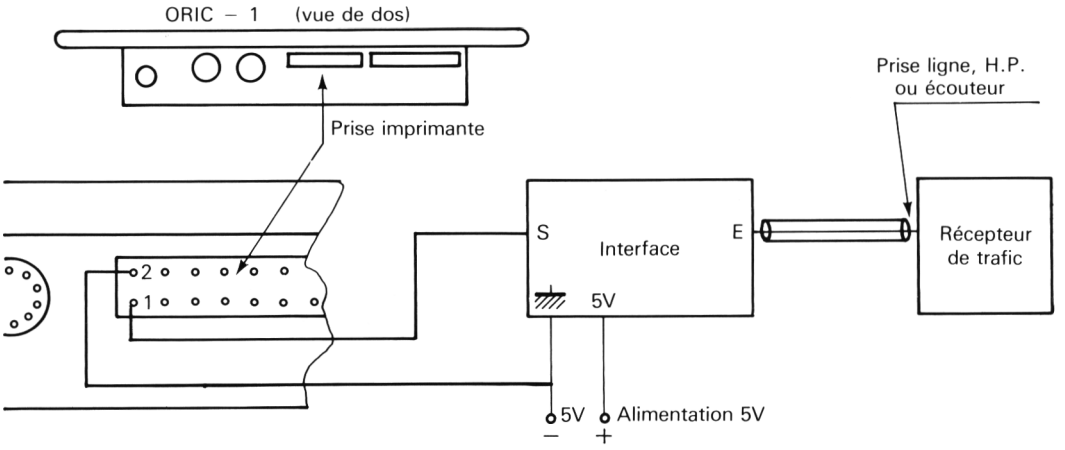
C1 0,22 μF
C2 0,1 μF
C3 1 μF
C4 0,1 μF
C5 47 μF - 10 V

IMPLANTATION DES COMPOSANTS





CIRCUIT IMPRIME



ORICMORSE — PARTIE BASIC

```
2 REM    ++++ORICMORSE++++
3 REM    +       @        +
4 REM    +BONOMO-DUTERTRE+
5 REM    + F6GKQ-F1EZH    +
6 REM    +    13-11-83    +
7 REM    +     (V.01)     +
8 REM    +  O R I C - 1   +
9 REM    ++++++++++++++++++
10 REM******ORICMORSE******
20 HIMEM#8000
30 FORN=#8009TO#81BC
40 READA:POKEN,A
50 NEXT
60 FORN=#8500TO#853E
70 READA:POKEN,A
80 NEXT
81 N=#820C
82 READA
83 IFA=#FETHEN85
84 POKEN,A:N=N+1:GOTO82
85 FORN=#8600TO#865A:POKEN,0:NEXT
87 FORN=#8629TO#865A:READA:POKEN,A:NEXT:POKE#8620,1
90 PRINTCHR$(12)
95 PRINTSPC(10)
100 PRINTCHR$(4);CHR$(27);"JORICMORSE";CHR$(4)
110 PLOT5,10,"E--EMISSION"
120 PLOT5,12,"R--RECEPTION"
125 PLOT5,14,"T--ENTRAINEMENT"
130 GETA$:IFA$=""THEN130
140 IFA$="R"THEN400
150 IFA$="E"THEN500
155 IFA$="T"THEN800
160 GOTO130
400 INK7:PAPER0:CLS
405 FORN=1TO25:PRINT:NEXT
410 PRINTCHR$(4);CHR$(27);"J          RETURN POUR EMETTRE";CHR$(4)
430 CALL#8009
```

```
435 POKE#26F,27:PAPER7:INK0:CLS
436 PLOT10,14,"EMISSION"
437 WAIT300
440 GOTO500
500 REM  EMISSION
505 A$="CQ CQ CQ CQ DE F1EZH..MESSAGE DE TEST MORSE AVEC ORIC.1 A L
 AIDE"
510 A$=A$+" DU PROGRAMME A DENIS F6GK0 ET EDDY F1EZH.CE MESSAGE NE
DOIT "
515 A$=A$+"AVOIR AU MAXIMUM QUE 254 CARACTERES.BON TRAFIC AVEC ORIC
.1"
516 CLS:PRINT"LE MESSAGE EN MEMOIRE EST :":PRINT:PRINTA$
520 IFLEN(A$)>254THENL=254ELSEL=LEN(A$)
525 FORN=1TOL:POKE#82FF+N,ASC(MID$(A$,N,1)):NEXT
530 POKE#82FF+N,62
535 PRINT:PRINT:PRINT
540 INPUT"COMMUTATION(O/N)";R$
545 IFR$="O"THENPOKE#45,7ELSEPOKE#45,247
550 INPUT"VITESSE(1 a 10)";V
555 IFV>10ORV<1THEN550
560 V=(11-V)*8
570 POKE#9200,V:POKE#9201,V*3:POKE#9202,V*2
573 CLS
575 PLOT1,21,"------------------------------------------"
576 POKE#26F,21
577 GOSUB700
580 SOUND1,90,0
590 PRINTCHR$(6):CALL#820D:PRINTCHR$(6)
595 POKE#26F,27:CLS
600 GOTO110
700 PLOT1,22,"TOUCHES CURSEUR:VOLUME"
710 PLOT1,24,"TOUCHE '#'      :MESSAGE MEMORISE"
720 PLOT1,26,"TOUCHE 'RETURN':RECEPTION"
730 RETURN
800 CLS
802 PLOT1,13,"PATIENTEZ JE COMPOSE LA DICTEE"
805 FORN=0TO253
810 G=INT(RND(1)*90)
815 IFG<33THEN810
820 IFPEEK(#9600+G)=0THEN970
830 POKE#8300+N,G
840 NEXT
845 POKE#8300+N,62:PING
```

```
847 CLS
850 INPUT"VITESSE(1 a 10)":V
855 V=(11-V)*8
860 POKE#8200,V:POKE#8202,V*2:POKE#8201,V*3
870 PRINT:PRINT"APPUYEZ SUR UNE TOUCHE POUR COMMENCER"
880 GETA$:IFA$=""THEN880
885 CLS
886 WAIT100:POKE#45,247
890 SOUND1,90,0:PAPER0:CALL#82D0
900 PAPER7
910 PLOT1,20,"D-POUR UNE AUTRE DICTEE"
920 PLOT1,22,"A-POUR ARRETER"
930 GETG$
940 IFG$="D"THEN800
950 IFG$="A"THENCLS:GOTO110
960 GOTO930
970 IFPEEK(#8300+N-1)=32THEN810
980 POKE#8300+N,32:NEXT
990 GOTO845
1000 REM***CODES MACHINE***
1010 DATA#20,#0E,#91,#A9,#00,#8D,#02,#03,#A9,#10,#8D,#00,#80,#A9
1020 DATA#15,#8D,#02,#80,#8D,#06,#80,#A9,#00,#8D,#03,#80,#8D,#04
1030 DATA#80,#8D,#05,#80,#EA,#AD,#00,#03,#29,#10,#D0,#F9,#AD,#00
1040 DATA#03,#29,#10,#F0,#F9,#98,#48,#A0,#07,#A2,#CA,#CA,#D0,#FD
1050 DATA#88,#D0,#F8,#68,#A8,#AD,#00,#03,#29,#10,#CD,#00,#90,#D0
1060 DATA#0D,#A9,#FF,#CD,#01,#90,#F0,#0E,#EE,#01,#80,#4C,#38,#80
1070 DATA#8D,#00,#80,#AD,#00,#80,#F0,#5B,#AD,#06,#80,#0A,#EA,#CD
1080 DATA#01,#80,#F0,#13,#10,#03,#4C,#82,#80,#0E,#05,#80,#0E,#04
1090 DATA#80,#A9,#00,#8D,#01,#80,#4C,#57,#80,#AD,#04,#80,#0A,#18
1100 DATA#6D,#05,#80,#AA,#BD,#00,#85,#AA,#20,#3A,#81,#AD,#00,#03
1110 DATA#29,#AF,#C9,#AF,#F0,#6B,#A9,#00,#8D,#04,#80,#8D,#05,#80
1120 DATA#AD,#06,#80,#0A,#0A,#0A,#EA,#EA,#CD,#01,#80,#F0,#08,#10
1130 DATA#03,#4C,#B8,#80,#4C,#A8,#81,#A2,#20,#20,#3A,#81,#4C,#A8
1140 DATA#81,#0E,#02,#80,#AD,#01,#80,#0A,#18,#6D,#01,#80,#CD,#02
1150 DATA#80,#F0,#25,#10,#03,#4C,#F5,#80,#AD,#01,#80,#CD,#02,#80
1160 DATA#F0,#02,#10,#05,#AD,#03,#80,#F0,#11,#A9,#01,#8D,#03,#80
1170 DATA#EE,#05,#80,#AD,#01,#80,#8D,#02,#80,#4C,#7A,#80,#AD,#01
1180 DATA#80,#8D,#06,#80,#A9,#00,#8D,#03,#80,#EE,#04,#80,#4C,#EC
1190 DATA#80,#A9,#F7,#8D,#02,#03,#58,#60,#EA,#78,#A9,#FF,#AA,#A9
1200 DATA#07,#20,#25,#F5,#A9,#00,#AA,#A9,#0E,#20,#35,#F5,#A0,#00
1210 DATA#A9,#2D,#99,#40,#BF,#C8,#98,#C9,#28,#D0,#F5,#A9,#00,#8D
1220 DATA#07,#80,#A8,#A9,#17,#8D,#6F,#02,#60,#EA,#EA,#AD,#07,#80
```

```
1230 DATA#F0,#20,#C9,#01,#F0,#32,#C9,#02,#F0,#3A,#B9,#A9,#BE,#C9
1240 DATA#2D,#F0,#3F,#EA,#EA,#EA,#EA,#EA,#8A,#99,#A8,#BE,#C8,#60
1250 DATA#A9,#00,#8D,#07,#80,#A8,#EA,#EA,#EA,#EA,#EA,#8A,#99,#A8
1260 DATA#BB,#C8,#98,#C9,#00,#F0,#01,#60,#EE,#07,#80,#A0,#00,#60
1270 DATA#EA,#EA,#EA,#EA,#EA,#8A,#99,#A8,#BC,#4C,#68,#81,#EA,#EA
1280 DATA#EA,#EA,#EA,#8A,#99,#A8,#BD,#4C,#68,#81,#8A,#48,#20,#0A
1290 DATA#CC,#A9,#FF,#AA,#A9,#07,#20,#35,#F5,#A9,#00,#AA,#A9,#0E
1300 DATA#20,#35,#F5,#68,#AA,#A9,#00,#F0,#B1,#AD,#07,#80,#F0,#03
1310 DATA#4C,#7A,#80,#98,#C9,#01,#D0,#F8,#A9,#02,#8D,#01,#80,#4C,#5
A,#80
1320 REM***TABLE DE TRANSCODAGE***
1330 DATA#20,#54,#45,#4D,#4E,#41,#49,#4F,#47,#4B,#44,#57,#52,#55
1340 DATA#53,#20,#20,#51,#5A,#59,#43,#58,#42,#4A,#50,#3A,#4C,#20
1350 DATA#46,#56,#48,#30,#39,#20,#38,#20,#20,#20,#37,#20,#3E,#2E
1360 DATA#20,#2C,#2F,#2D,#36,#31,#20,#22,#3F,#20,#2A,#20,#23,#32
1370 DATA#20,#5B,#20,#33,#20,#34,#35
1380 REM***EMISSION***
1390 DATA#60,#20,#05,#E9,#C9,#00,#F0,#F9,#C9,#0D,#F0,#F4,#AA,#20
1400 DATA#A6,#82,#BD,#00,#86,#20,#27,#82,#4C,#0D,#82,#EA,#EA
1410 DATA#C9,#01,#F0,#5C,#78,#A0,#08,#8C,#03,#82,#2A,#B0,#0C,#CE
1420 DATA#03,#82,#D0,#F8,#58,#60,#EA,#EA,#EA,#EA,#EA,#CE,#03,#82
1430 DATA#F0,#39,#2A,#B0,#1F,#A2,#05,#48,#A9,#08,#20,#94,#82,#AE
1440 DATA#00,#82,#20,#75,#82,#A2,#00,#A9,#08,#20,#9D,#82,#AE,#00
1450 DATA#82,#20,#75,#82,#68,#4C,#40,#82,#A2,#05,#48,#A9,#08,#20
1460 DATA#94,#82,#AE,#01,#82,#4C,#53,#82,#A0,#FF,#98,#D0,#FD,#CA
1470 DATA#D0,#F8,#60,#AE,#01,#82,#20,#75,#82,#4C,#39,#82,#AE,#01,#8
2,#20
1480 DATA#75,#82,#AE,#02,#82,#20,#75,#82,#60,#20,#35,#F5,#A5,#45
1490 DATA#8D,#02,#03,#60,#20,#35,#F5,#A9,#F7,#8D,#02,#03,#60,#C9
1500 DATA#0A,#F0,#12,#C9,#0B,#D0,#1A,#AD,#49,#82,#C9,#0F,#F0,#06
1510 DATA#EE,#49,#82,#EE,#68,#82,#60,#AD,#49,#82,#F0,#06,#CE,#49
1520 DATA#82,#CE,#68,#82,#60,#C9,#23,#F0,#04,#20,#3F,#F7,#60,#A9
1530 DATA#00,#8D,#05,#82,#AD,#05,#82,#AA,#BD,#00,#83,#C9,#3E,#F0,#1
0
1540 DATA#AA,#20,#3F,#F7,#BD,#00,#86,#20,#27,#82,#EE,#05,#82,#4C
1550 DATA#D5,#82,#A9,#00,#A2,#00,#60,#FE,#2D,#00,#00,#28,#00,#35
1560 DATA#32,#2F,#2F,#27,#23,#21,#20,#30,#38,#3C,#3E,#00,#00,#00
1570 DATA#00,#00,#4C,#00,#05,#19,#1A,#0C,#02,#12,#0E,#10,#04,#17
1580 DATA#0D,#14,#07,#06,#0F,#16,#1D,#0A,#08,#03,#09,#11,#0B,#19
1590 DATA#1B,#1C
```

```
9D
8009 200E81 JSR #810E      Routine d'initiation
800C A900   LDA %#00
800E 8D0203 STA #0302      Programme le VIA en entrée
8011 A910   LDA %#10
8013 8D0080 STA #8000      L = 10
8016 A915   LDA %#15
8018 8D0280 STA #8002      DP = DR = 15
801B 8D0680 STA #8006
801E A900   LDA %#00
8020 8D0280 STA #8003      T = 0 (point)
8023 8D0480 STA #8004
8026 8D0580 STA #8005
8029 EA     NOP
802A AD0003 LDA #0300      Attente d'un espace
802D 2910   AND %#10
802F D0F9   BNE #802A
8031 AD0003 LDA #0300      Attente d'un élément
8034 2910   AND %#10
8036 F0F9   BEQ #8031
8038 98     TYA            Préserve le contexte
8039 48     PHA
803A A007   LDY %#07       Temporisation pour un échantillonnage à 5 ms
803C A2CA   LDX %#CA
803E CA     DEX
803F D0FD   BNE #803E
8041 88     DEY
8042 D0F8   BNE #803C
8044 68     PLA            Restitue le contexte
8045 A8     TAY
8046 AD0003 LDA #0300
8049 2910   AND %#10       Lecture du niveau d'entrée et comparaison avec L
804B CD0080 CMP #8000
804E D00D   BNE #805D
8050 A9FF   LDA %#FF
8052 CD0180 CMP #8001      Boucle si identique
8055 F00E   BEQ #8065
8057 EE0180 INC #8001
805A 4C3880 JMP #8038
805D 8D0080 STA #8000      Sinon, mémorise
8060 AD0080 LDA #8000
8063 F05B   BEQ #80C0
8065 AD0680 LDA #8006      Elément ou Espace ?
```

```
8068 0A      ASL A
8069 EA      NOP
806A CD0190  CMP #8001
806D F013    BEQ #8082
806F 1003    BPL #8074
8071 4C8280  JMP #8082
8074 0E0580  ASL #8005
8077 0E0480  ASL #8004
807A A900    LDA %#00
807C 8D0190  STA #8001
807F 4C5780  JMP #8057
8082 AD0480  LDA #8004
8085 0A      ASL A
8086 18      CLC
8087 6D0580  ADC #8005
808A AA      TAX
808B BD0085  LDA #8500,X
808E AA      TAX
808F 203A81  JSR #813A
8092 AD0003  LDA #0300
8095 29AF    AND %#AF
8097 C9AF    CMP %#AF
8099 F06B    BEQ #8106
809B A900    LDA %#00
809D 8D0480  STA #8004
80A0 8D0580  STA #8005
80A3 AD0680  LDA #8006
80A6 0A      ASL A
80A7 0A      ASL A
80A8 0A      ASL A
80A9 EA      NOP
80AA EA      NOP
80AB CD0190  CMP #8001
80AE F008    BEQ #80B8
80B0 1003    BPL #80B5
80B2 4CB880  JMP #80B8
80B5 4CA881  JMP #81A8
80B8 A220    LDX %#20
80BA 203A81  JSR #813A
80BD 4CA881  JMP #81A8
80C0 0E0280  ASL #8002
80C3 AD0190  LDA #8001
80C6 0A      ASL A
```

A = 2P

DR 2P

Transcodage MORSE ASCII

Saut vers Affichage

Test du clavier (Touche RETURN)

Remise à zéro des variables « trait » et « point »

A = 4P

Test espace inter-mots

Affichage d'un Espace

DP = 2 DP

```
80C7 18      CLC
80C8 6D0180  ADC #8001     A = 3 DR
80CB CD0280  CMP #8002
80CE F025    BEQ #80F5
80D0 1003    BPL #80D5
80D2 4CF580  JMP #80F5
80D5 AD0180  LDA #8001     DR    2 DP
80D8 CD0280  CMP #8002
80DB F002    BEQ #80DF
80DD 1005    BPL #80E4
80DF AD0380  LDA #8003
80E2 F011    BEQ #80F5
80E4 A901    LDA %#01      Cas d'un trait
80E6 8D0380  STA #8003
80E9 EE0580  INC #8005
80EC AD0180  LDA #8001
80EF 8D0280  STA #8002
80F2 4C7A80  JMP #807A
80F5 AD0180  LDA #8001
80F8 8D0680  STA #8006
80FB A900    LDA %#00
80FD 8D0380  STA #8003
8100 EE0480  INC #8004
8103 4CEC80  JMP #80EC
8106 A9F7    LDA %#F7      Prépare le retour au BASIC
8108 8D0203  STA #0302
810B 58      CLI
810C 60      RTS           Retour au BASIC
810D EA      NOP
810E 78      SEI
810F A9FF    LDA %#FF      Prépare la lecture du clavier
8111 AA      TAX
8112 A907    LDA %#07
8114 2035F5  JSR #F535
8117 A900    LDA %#00
8119 AA      TAX
811A A90E    LDA %#0E
811C 2035F5  JSR #F535
811F A000    LDY %#00      Trace ligne de pointillés à la 18e ligne
8121 A92D    LDA %#2D
8123 9940BF  STA #BF40,Y
8126 C8      INY
8127 98      TYA
```

```
8128 C928   CMP %#28
812A D0F5   BNE #8121
812C A900   LDA %#00
812E 8D0780 STA #8007
8131 A8     TAY
8132 A917   LDA %#17
8134 8D6F02 STA #026F
8137 60     RTS
8138 EA     NOP
8139 EA     NOP
813A AD0780 LDA #8007
813D F020   BEQ #815F
813F C901   CMP %#01
8141 F032   BEQ #8175
8143 C902   CMP %#02
8145 F03A   BEQ #8181
8147 B9A8BE LDA #BEA8,Y
814A C92D   CMP %#2D
814C F02F   BEQ #819D
814E EA     NOP
814F EA     NOP
8150 EA     NOP
8151 EA     NOP
8152 EA     NOP
8153 8A     TXA
8154 99A8BE STA #BEA8,Y
8157 C8     INY
8158 60     RTS
8159 A900   LDA %#00
815B 8D0780 STA #8007
815E A8     TAY
815F EA     NOP
8160 EA     NOP
8161 EA     NOP
8162 EA     NOP
8163 EA     NOP
8164 8A     TXA
8165 99A8BB STA #BBA8,Y
8168 C8     INY
8169 98     TYA
816A C900   CMP %#00
816C F001   BEQ #816F
816E 60     RTS
```

Fenêtre écran de 17 lignes

ROUTINE D'AFFICHAGE

```
816F EE0780 INC #8007
8172 A000   LDY %#00
8174 60     RTS
8175 EA     NOP
8176 EA     NOP
8177 EA     NOP
8178 EA     NOP
8179 EA     NOP
817A 8A     TXA
817B 99A8BC STA #BCA8,Y
817E 4C6881 JMP #8168
8181 EA     NOP
8182 EA     NOP
8183 EA     NOP
8184 EA     NOP
8185 EA     NOP
8186 8A     TXA
8187 99A8BD STA #BDA8,Y
818A 4C6881 JMP #8168
818D 8A     TXA
818E 48     PHA
818F 200ACC JSR #CC0A   Effacement de l'écran
8192 A9FF   LDA %#FF
8194 AA     TAX         Prépare lecture du clavier
8195 A907   LDA %#07
8197 2035F5 JSR #F535
819A A900   LDA %#00
819C AA     TAX
819D A90E   LDA %#0E
819F 2035F5 JSR #F535
81A2 68     PLA
81A3 AA     TAX
81A4 A900   LDA %#00
81A6 F0B1   BEQ #8159   Retour en haut d'écran
81A8 AD0780 LDA #8007
81AB F003   BEQ #81B0
81AD 4C7A80 JMP #807A
81B0 98     TYA
81B1 C901   CMP %#01
81B3 D0F8   BNE #81AD
81B5 A902   LDA %#02
81B7 8D0180 STA #8001
81BA 4C5A80 JMP #805A
```

S/P Emission

```
820C 60      RTS
820D 2005E9  JSR #E905
8210 C900    CMP %#00
8212 F0F9    BEQ #820D
8214 C90D    CMP %#0D
8216 F0F4    BEQ #820C
8218 AA      TAX
8219 20A682  JSR #82A6
821C BD0086  LDA #8600,X
821F 202782  JSR #8227
8222 4C0D82  JMP #820D
8225 EA      NOP
8226 EA      NOP
8227 C901    CMP %#01
8229 F05C    BEQ #8287
822B 78      SEI
822C A008    LDY %#08
822E 8C0382  STY #8203
8231 2A      ROL A
8232 B00C    BCS #8240
8234 CE0382  DEC #8203
8237 D0F8    BNE #8231
8239 58      CLI
823A 60      RTS
823B EA      NOP
823C EA      NOP
823D EA      NOP
823E EA      NOP
823F EA      NOP
8240 CE0382  DEC #8203
8243 F039    BEQ #827E
8245 2A      ROL A
8246 B01F    BCS #8267
8248 A20A    LDX %#0A
824A 48      PHA
824B A908    LDA %#08
824D 209482  JSR #8294
8250 AE0082  LDX #8200
8253 207582  JSR #8275
```

Retour au basic

Lecture clavier (DÉBUT).

Si non appui touche, saut à lecture clavier

Si appui touche « RETURN », retour au basic

X contient le code ASCII de la touche

Saut au sous-programme de test touches spéciales

Transcodage ASCII → MORSE

Saut à la routine « SON »

Boucle à lecture clavier

Inhibe les interruptions

Nb de bits

Attente bit de « start »

Si bit de start, saut en 8240

Boucle si non bit de start

Retour du sous-programme « SON »

Décompte le bit de start

Si le caractère est fini, saut à traitement espace inter-caractères

Prend le bit suivant

Si bit = 1 saut au traitement trait

Génère tonalité

Tempo d'un point

```
8256 A200    LDX %#00
8258 A908    LDA %#08
825A 209D82  JSR #829D      Arrête tonalité
825D AE0082  LDX #8200
8260 207582  JSR #8275      Génère silence pendant 1 point
8263 68      PLA
8264 4C4082  JMP #8240      Continu
8267 A20A    LDX %#0A
8269 48      PHA
826A A908    LDA %#08
826C 209482  JSR #9294      Génère tonalité
826F AE0182  LDX #8201
8272 4C5382  JMP #8253      Tempo d'un trait et boucle
8275 A0FF    LDY %#FF
8277 88      DEY
8278 D0FD    BNE #8277      Sous-programme de temporisation
827A CA      DEX
827B D0F8    BNE #8275
827D 60      RTS
827E AE0182  LDX #8201
8281 207582  JSR #8275      Tempo inter-caractère
8284 4C3982  JMP #8239      Retour lecture clavier
8287 AE0182  LDX #8201
828A 207582  JSR #8275
828D AE0282  LDX #8202
8290 207582  JSR #8275
8293 60      RTS
8294 2035F5  JSR #F535      Autorise le générateur sonore
8297 A545    LDA #45        Colle le relai
8299 8D0203  STA #0302
829C 60      RTS            Retour
829D 2035F5  JSR #F535      Inhibe le générateur sonore
82A0 A9F7    LDA %#F7       Décolle le relai
82A2 8D0203  STA #0302
82A5 60      RTS            Retour
82A6 C90A    CMP %#0A       Lecture des touches spéciales
82A8 F012    BEQ #82BC      Si touche « ↓ » saut au S/P de diminution du niveau sonore
82AA C90B    CMP %#0B
82AC D01A    BNE #82C8      Si touche « ↑ » saut au S/P d'augmentation niveau sonore
82AE AD4982  LDA #8249
82B1 C90F    CMP %#0F       Retour si niveau maxi
82B3 F006    BEQ #82BB
82B5 EE4982  INC #8249
```

```
82B9 EE6882 INC #8268
82BB 60     RTS
82BC AD4982 LDA #8249
82BF F006   BEQ #82C7
82C1 CE4982 DEC #8249
82C4 CE6882 DEC #8268
82C7 60     RTS
82C9 C923   CMP %#23
82CA F004   BEQ #82D0
82CC 203FF7 JSR #F73F
82CF 60     RTS
82D0 A900   LDA %#00
82D2 8D0582 STA #8205
82D5 AD0582 LDA #8205
82D8 AA     TAX
82D9 BD0083 LDA #8300,X
82DC C93E   CMP %#3E
82DE F010   BEQ #82F0
82E0 AA     TAX
82E1 203FF7 JSR #F73F
82E4 BD0086 LDA #8600,X
82E7 202782 JSR #8227
82EA EE0582 INC #8205
82ED 4CD582 JMP #82D5
82F0 A900   LDA %#00
82F2 A200   LDX %#00
82F4 60     RTS
82F5 5555   EOR #55,X
82F7 5555   EOR #55,X
```

Augmente le niveau
Retour

Retour si niveau mini

Baisse le niveau
Retour

Si touche « # » saut au message mémorisé

Sinon affiche caractère et retour

Initialise lecture au début du message

Lecture caractère

Arrêt et retour si caractère fin de message

Affiche caractère

Transcodage ASCII → MORSE

Emission caractère morse

Prend le caractère suivant et boucle

Retour du S/P message mémorisé

PROGRAMME RTTY

Le micro-ordinateur peut, nous le savons maintenant, faire beaucoup de choses et, dans le domaine des communications, il est tout-à-fait concevable de l'utiliser en émission-réception radio-télétype (RTTY).

Nous avons donc développé un logiciel capable de transformer votre ORIC-1 à cet effet. Il est important de souligner que le couplage du micro-ordinateur au démodulateur RTTY et direct. Cela signifie que, si vous possédez déjà une machine mécanique, le démodulateur (à filtres actifs ou à P.L.L.) sera utilisable et se connectera directement sur l'ORIC.

CARACTÉRISTIQUES DU PROGRAMME

Le couplage de la station au micro-ordinateur s'effectue par le biais de la prise imprimante de ce dernier.

Le programme autorise alors l'émission et la réception des signaux RTTY. Les vitesses retenues sont 45.45 bauds et 50 bauds qui sont les plus fréquemment utilisés. Le shift à l'émission est programmable. Le shift en réception se sélectionne, bien sûr, sur le démodulateur.

PARTIE ÉMISSION :

a) La vitesse et le shift sont donc réglables.

b) L'émission du retour chariot (RC) (carriage return (CR) pour les anglais) est automatique ou non. Si vous choisissez le mode automatique, le programme va générer, automatiquement, un retour chariot toutes les N colonnes. Dans ce cas, c'est vous qui allez définir, à l'initialisation, ce nombre N (par exemple 32 ou 40 ou 64 colonnes, ou toute autre valeur).

Ce nombre est surtout fonction du type de matériel utilisé en réception, chez votre correspondant. Comme dans le mode « manuel » l'appui sur RETURN provoquera néanmoins l'émission d'un retour-chariot.

Le mode « manuel » s'obtient en demandant le retour-chariot après... zéro (0) colonnes.

Le retour-chariot est émis en compagnie d'une avance papier.

Cette option avec ou sans retour chariot permet une grande souplesse d'utilisation. Par exemple, si deux ORIC sont en liaison, il ne sera pas nécessaire d'émettre les Retours-chariot automatiques. C'est même déconseillé pour la présentation, à moins de choisir d'émettre ces R.C. avant la 40e colonne.

c) L'émission peut se faire en mode « clavier » ou en mode « mémoire ».

En mode clavier, les caractères sont émis au fur et à mesure que vous appuyez sur les touches.

En mode mémoire, des textes peuvent être prédéfinis, qui contiendront, par exemple, un message d'appel, vos conditions de trafic etc...

On peut passer du mode clavier au mode mémoire en appuyant sur la touche « » (shift 6).

Ces changements d'option permettent de mélanger les modes clavier et mémorisés.

PARTIE RÉCEPTION :

En réception, le programme fonctionne en 45.45 ou 50 bauds.

Plusieurs options sont disponibles, pendant le décodage :

a) forçage de modes : il peut arriver que l'on se synchronise mal sur un message à recevoir, parce qu'on le prend en cours, ou qu'on perde la synchronisation à cause d'un coup de fadding. On se retrouve alors avec des séries de chiffres ou signes (ou de lettres si le message était une succession de chiffres). Il est possible de forcer la machine en chiffres ou en lettres,

b) effaçage de l'écran : on efface le contenu de la partie de l'écran réservée au texte décodé. Ceci provoque le retour du curseur en haut, à gauche de l'écran,

c) affichage mémoire : au fur et à mesure que le message est décodé, il est mémorisé. On dispose ainsi des 4095 derniers caractères reçus... Il est possible d'afficher le contenu de cette mémoire. Le texte défilera alors à l'écran assez rapidement et s'arrêtera pendant tout le temps où vous maintiendrez la touche espace,

d) passage en émission : deux possibilités ont été prévues selon que vous souhaitez ou non, modifier les paramètres shift et vitesse précédemment définis.

Soulignons enfin que le passage émission-réception s'accompagne de la télécommande de la station, si vous prenez soin toutefois de câbler la liaison sur la prise « magnétophone » de l'ORIC comme indiqué plus loin.

FONCTIONNEMENT AUTOMATIQUE

Nous avons vu précédemment (programme « Fréquence ») que l'ORIC était capable de mesurer la fréquence d'un signal BF introduit sur sa prise lecture K7.

Nous avons utilisé cette propriété pour créer un mode de fonctionnement automatique de la machine. Ainsi, il sera possible de consulter, en l'absence de l'opérateur, la machine si celle-ci a été placée en mode « VEILLE » (option A du « menu »).

Le principe retenu est simple. Dans ce mode, ORIC, attend une tonalité à une fréquence prédéfinie par les utilisateurs du système. Ici, cette fréquence est à 2500 Hz mais le programme peut être modifié, et la fréquence de commande remplacée par toute autre valeur

située dans la bande BF du récepteur (300-3000 Hz). Attention néanmoins au choix de cette fréquence ; il faudra éviter par exemple les fréquences MARK et SPACE ou le 1750 Hz si on ne veut pas que le système se déclenche intempestivement.

La machine attend donc de recevoir la tonalité de commande pendant environ 3 secondes et passe ensuite en émission. Elle transmettra au correspondant le message qu'on lui aura chargé au préalable en mémoire, message situé dans les lignes de DATA commençant en 10000, qu'il faudra donc modifier à chaque introduction d'un nouveau message, avant de lancer le programme. Ce message pourra contenir des informations de trafic, des prévisions de passages de satellites, des données diverses. Veillez néanmoins à ne pas sortir des conditions autorisées par la licence d'exploitation.

Si le correspondant utilise le même programme, il utilisera l'option T du « menu » pour générer la tonalité de commande, puis devra taper « R » lorsque l'émission automatique aura commencé, pour la décoder.

A la fin de l'émission automatique du message mémorisé (qui doit se terminer obligatoirement par puis !) la machine qui l'a émis va repasser en attente pendant une dizaine de secondes.

Si elle reçoit, dans cet intervalle de temps la fréquence de commande émise par le demandeur, elle se placera en réception et le demandeur pourra laisser un message. Ce message devra obligatoirement se terminer par un « ! ». La machine conservera le message en mémoire. Son propriétaire pourra le consulter. Après réception du « ! », la machine repasse en veille et le cycle pourra recommencer...

On dispose ainsi d'une sorte de boîte aux lettres électronique, capable d'émettre et de recevoir des messages. Ce système fonctionne de façon très fiable en VHF après définition d'une fréquence de veille et de la tonalité de commande.

Le couplage au récepteur s'effectue sur la prise BF (HP supplémentaire ou RECORD, ou autre...) qui sera reliée à la fois au démodulateur RTTY et à la prise « LOAD » de l'ORIC, après modification du cordon. La sortie « logique » du démodulateur est toujours reliée à la prise imprimante de l'ORIC (voir plus loin).

RAPPELS SUR LE RTTY

Une liaison radio-télétype utilisant le code BAUDOT est une transmission sérielle sur 7 bits.

1 bit de départ (START)
5 bits de donnée
1 bit de fin (STOP)

Avec 5 bits de donnée on pourra donc coder nos 31 caractères. La vitesse de 45.45 bauds correspond à une durée de bit de 22.22 ms. La vitesse de 50 bauds correspond à une durée de bit de 20 ms.

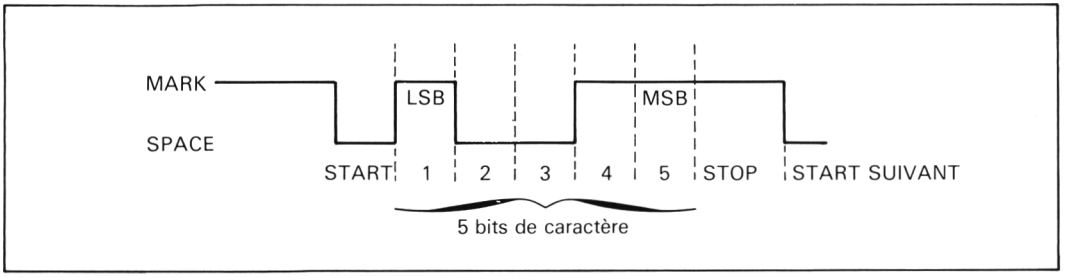
Les 2 états (haut-bas, 1 ou 0, MARK ou SPACE) sont codés à l'aide de 2 fréquences. L'écart entre ces 2 fréquences est le shift.

Ainsi avec une fréquence MARK à 1 275 Hz on a :

fréquence SPACE 1 445 Hz pour shift 170 Hz
1 700 Hz pour shift 425 Hz
2 125 Hz pour shift 850 Hz

Le rôle du démodulateur est de transformer ces signaux modulés en états 1 ou 0.

FORMAT D'UN CARACTÈRE EN CODE BAUDOT



Le bit de START est émis, suivi du bit de poids faible (LSB) du code du caractère etc... L'émission est close par le bit de STOP (qui dure 1,5 fois la durée des autres bits).

Ce découpage en éléments est appelé « moment ».

Le bit de START est un zéro (état bas, SPACE), le bit de STOP est un un (état haut, MARK).

Le code de la donnée représenté ci-dessus est donc 10011. (Il correspond à la lettre B ou au signe ?).

Un caractère spécial permet de passer des lettres aux chiffres et signes et réciproquement.

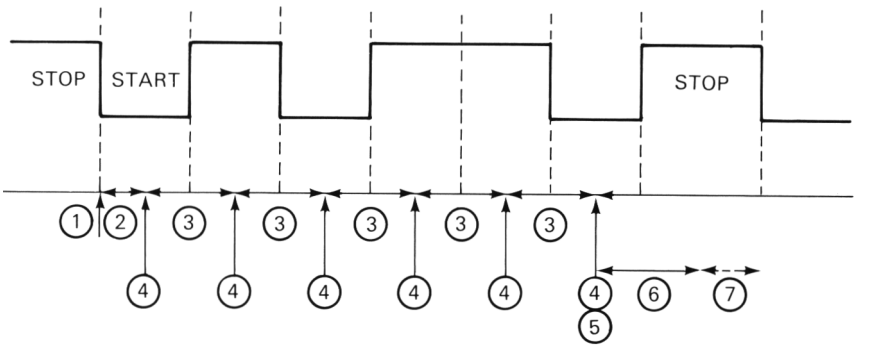
Le schéma ci-dessus nous permet de déduire le chronogramme de principe qui sera adopté par le programme pour effectuer le travail de décodage.

PRINCIPE DU PROGRAMME RTTY

1. CHRONOGRAMME DU DÉCODAGE

1 reconnaissance du bit de START (attente niveau 0)
2 délai 1/2 moment pour se centrer sur le milieu du bit
3 délai 1 moment pour se centrer sur le bit suivant
4 phase de lecture de l'état (0 ou 1) du moment
5 lecture du dernier moment
6 commence l'affichage et la mémorisation
7 attend la fin du bit de STOP et recommence la phase 1 .

1. Chronogramme du décodage



(1) Reconnaissance du bit de START (attente niveau 0)
(2) Délai 1/2 moment pour se centrer sur le milieu du bit
(3) Délai 1 moment pour se centrer sur le bit suivant
(4) Phase de lecture de l'état (0 ou 1) du moment.
(5) Lecture du dernier moment.
(6) Commence l'affichage et la mémorisation.
(7) Attend la fin du bit de STOP et recommence la phase (1).

2. RECONSTITUTION DU CODE DU CARACTÈRE

Le caractère est débarrassé des bits de START et de STOP.

on ne retient que les 5 bits de donnée. Ces 5 bits sont entrés tour à tour, après lecture de leur état, dans un registre et décalés au fur et à mesure, par la droite. Après lecture du 5e moment, on retrouve donc le bit de poids faible de la donnée en position du bit de poids faible dans le registre. Le caractère est reconstitué.

3. TRANSCODAGE

Le code ainsi obtenu ne peut être exploité directement, le micro-ordinateur ne reconnaissant que le code ASCII. On utilisera, pour établir la correspondance entre les deux, une table de transcodage où les éléments seront rangés par ordre croissant de leur code BAUDOT. Ainsi, par exemple, à la position 7 de la table (7 est la valeur baudot de la lettre U) on trouvera la valeur 55 qui est le code (hexadécimal) ASCII de cette même lettre.

Une fois le caractère lu dans la table, il est exploité pour être affiché sur l'écran.

4. ÉMISSION

A l'émission c'est la procédure inverse qui est utilisé. On lit le clavier. On établit la relation entre la touche pressée et le code BAUDOT correspondant grâce à la table de transcodage émission. Cette fois, les caractères y sont rangés par ordre croissant de leur code ASCII.

Le caractère est émis en utilisant, que le codage des deux tonalités MARK et SPACE, le générateur sonore inclus dans l'ORIC.

5. ÉMISSION DES MÉMOIRES

Le principe est strictement identique. Il faut seulement savoir que, au lieu de lire le calvier, on découpe les chaînes, représentant les messages mémorisés, caractère par caractère.

ARCHITECTURE DU PROGRAMME

Le programme utilise très largement des routines en langage machine. Elles seules peuvent garantir la rapidité requise. De plus, certaines fonctions ne seraient pas réalisables en restant sous BASIC.

L'organisation matérielle du programme est assez modulaire, ce qui permet d'éventuelles interventions pour modification.

Une zone mémoire importante a été dégagée pour implanter les routines en langage machine et les messages mémorisés, à l'émission comme à la réception (par moins de 4 K octets réservés pour la mémoire texte reçu !).

Pour cela, le plafond de mémoire utilisable par BASIC a été fixé à l'adresse # 7000.

Le programme BASIC est divisé en plusieurs parties :

- implantation des routines en langage machine
- DATA contenant le langage machine
- initialisations
- présentation et choix des paramètres
- partie « réception »
- partie « émission »
- le fonctionnement en automatique
- la génération de la tonalité d'ouverture
- les messages mémorisés
- le texte mémorisé, utilisé en mode automatique.

CODE BAUDOT

MODE LETTRE	MODE CHIFFRE		
A	—	11000	3
B	?	10011	25
C	:	01110	14
D	*	10010	9
E	3	10000	1
F	É	10110	13
G	%	01011	26
H	H	00101	20
I	8	01100	6
J	sonnerie	11010	11
K	(	11110	15
L	)	01001	18
M	.	00111	28
N	,	00110	12
O	9	00011	24
P	0	01101	22
Q	1	11101	23
R	4	01010	10
S	'	10100	5
T	5	00001	16
U	7	11100	7
V	=	01111	30
W	2	11001	19
X	/	10111	29
Y	6	10101	21
Z	+	10001	17
Retour chariot		00010	8
Saut de Ligne		01000	2
Préfixe Lettre		11111	31
Préfixe Chiffre		11011	27
Espace		00100	4

Le profil binaire correspondant est rétabli en décimal dans la dernière colonne du tableau (attention, on commence par la gauche pour calculer la valeur décimale).

Cette valeur décimale sera additionnée à l'offset chiffre/lettre pour trouver le caractère ASCII à l'emplacement correspondant de la table de transcodage réception.

A l'inverse, pour l'émission, les caractères seront rangés par ordre de code ASCII...

VARIABLES SYSTÈME UTILISÉES

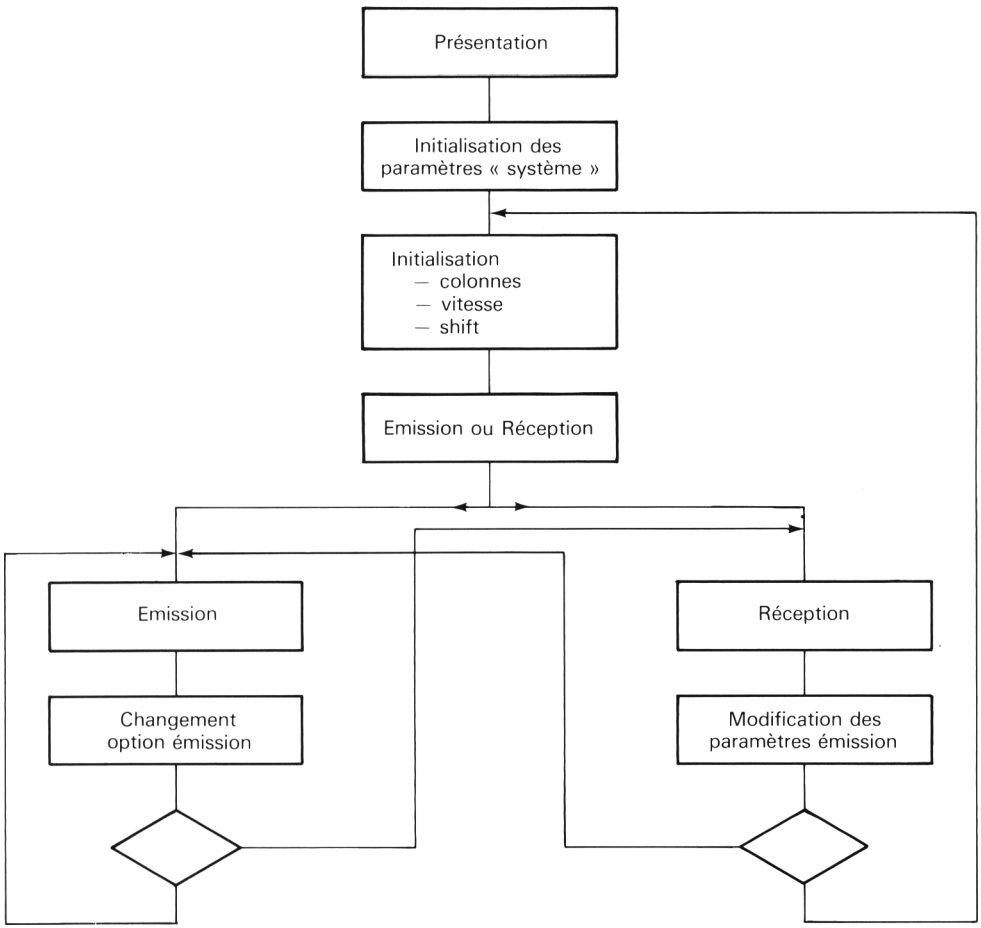
Bon nombre de variables système, définies dans le chapitre qui leur est consacré, ont été utilisées dans ce programme.

208 : contient code fonction de la touche clavier actionnée,
268 : numéro de ligne pour l'impression qui suit,
269 : numéro de colonne pour l'impression qui suit,
26A : pour forcer ORIC en 40 colonnes et avec clavier muet,
26F : pour n'utiliser qu'une fenêtre de 10 lignes, située en haut de l'écran. La partie inférieure affiche en permanence les options selon le mode. Cette fenêtre « restreinte » a son importance en réception car la routine qui affiche et effectue le scrolling est assez lente. Elle est fonction du nombre de lignes. Au-delà de 11, on perd des caractères, en 50 bds, lors du scrolling.

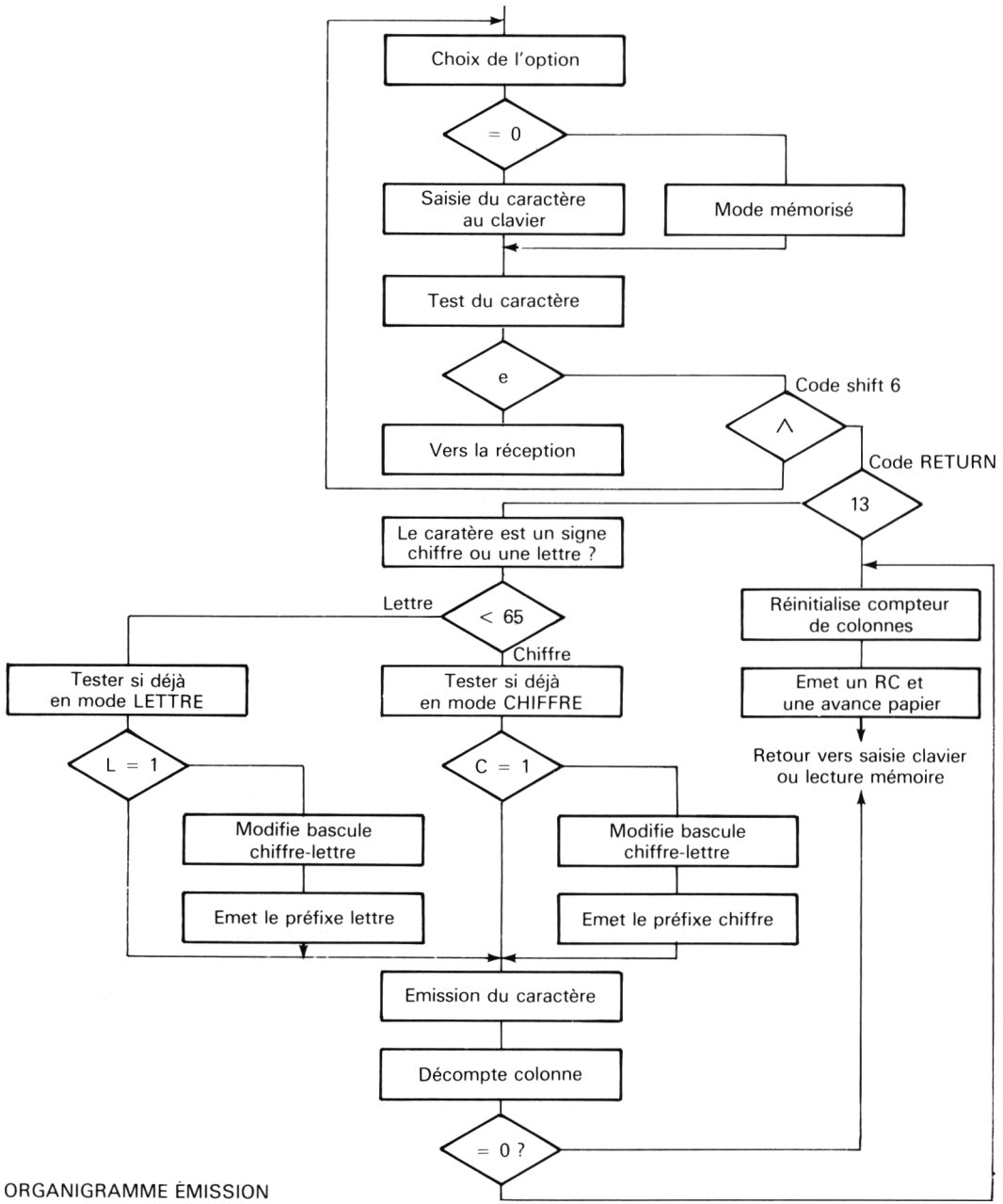
L'espace mémoire réservé aux adresses d'entrées-sorties a été utilisé, avec le VIA.

300 : registre de sortie du port B,
302 : sens des lignes du port B,
308 : octet de poids faible du TIMER 2 du VIA,
309 : octet de poids fort du TIMER 2,
30D : registre IFR.

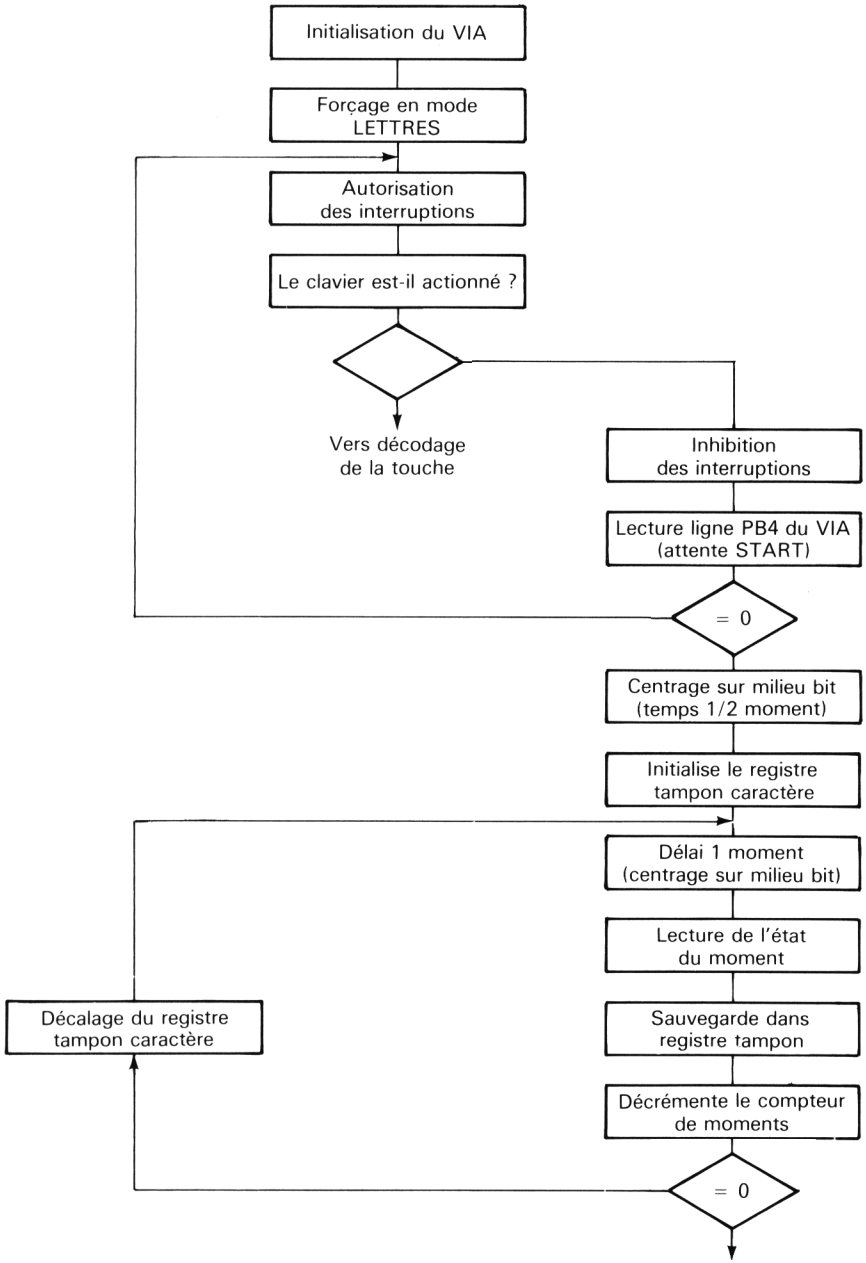
La télécommande émission-réception est également assurée à l'aide du relais interne servant habituellement au moteur du magnétophone. Attention à ne pas faire passer trop de courant par les contacts de ce relais !



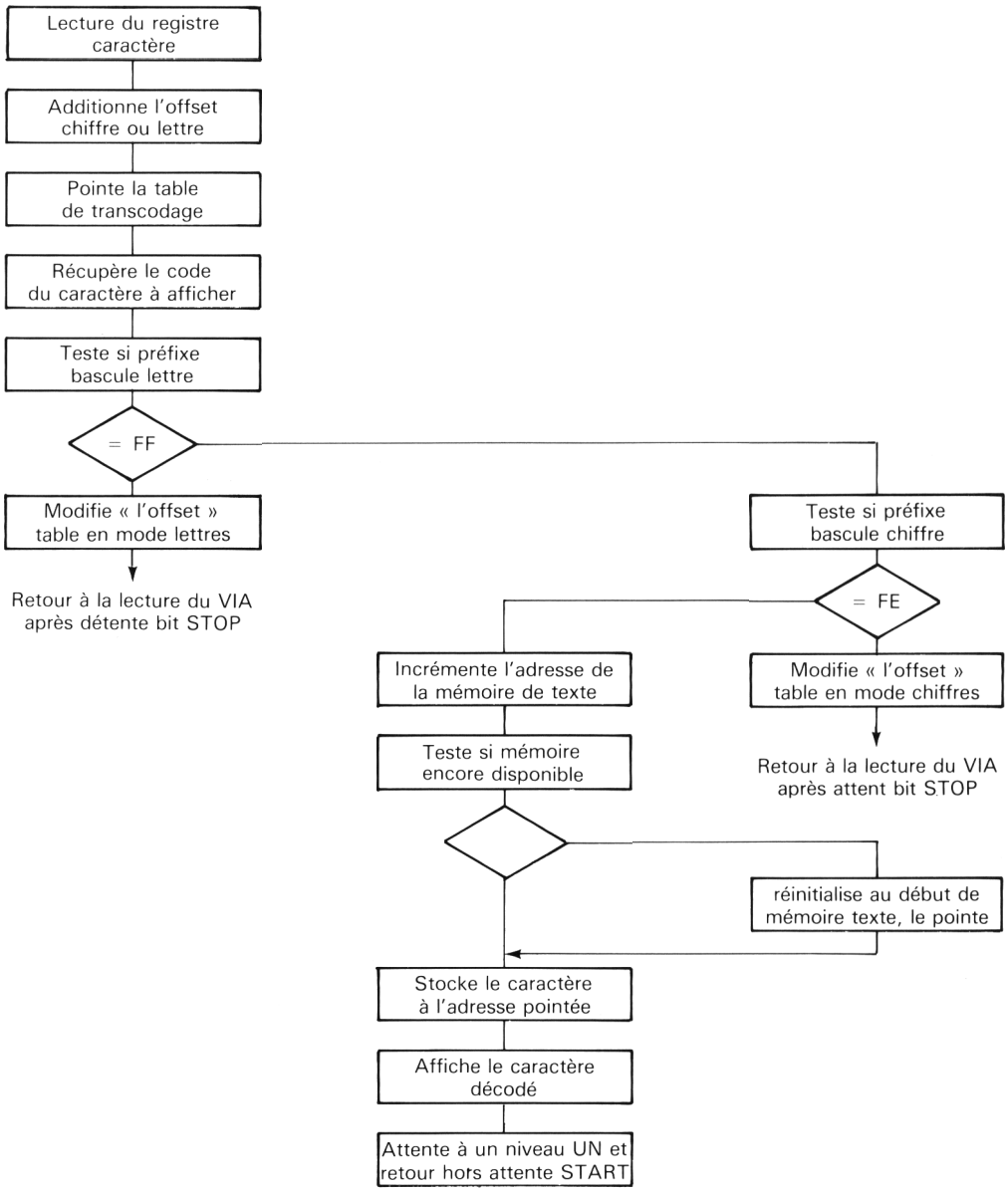
ORGANIGRAMME GÉNÉRAL

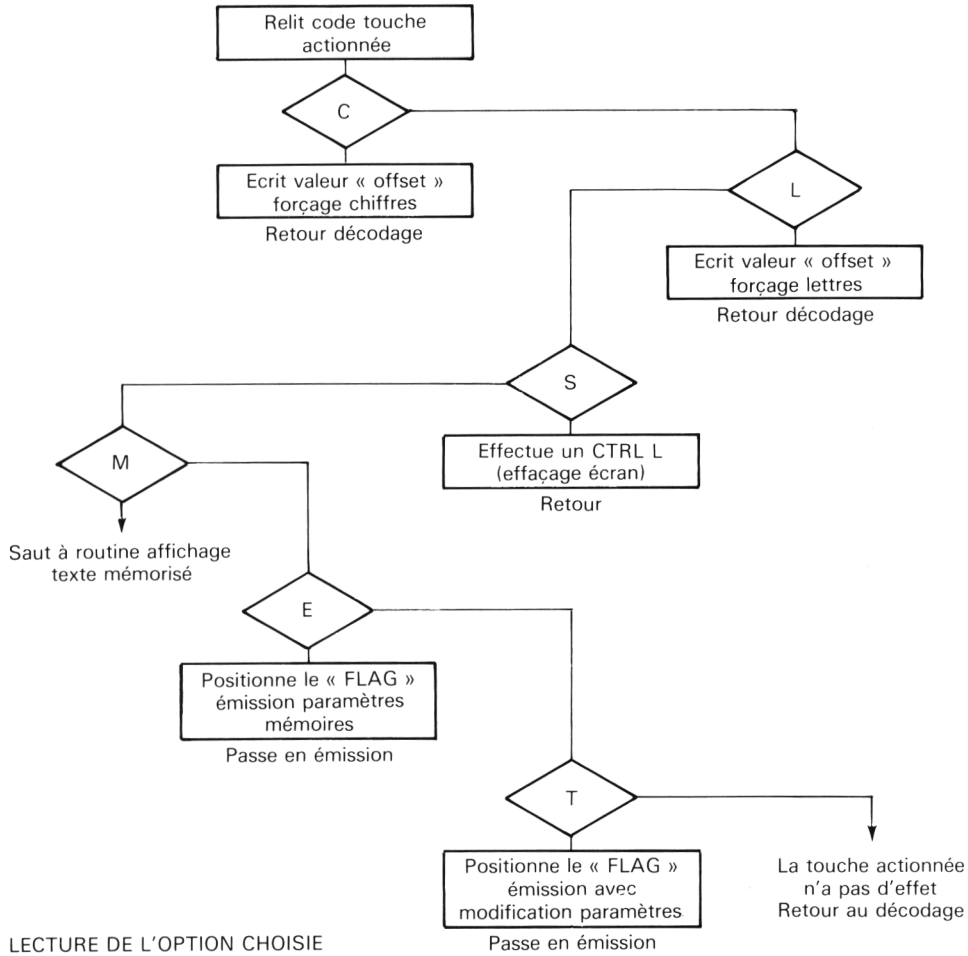


ORGANIGRAMME ÉMISSION

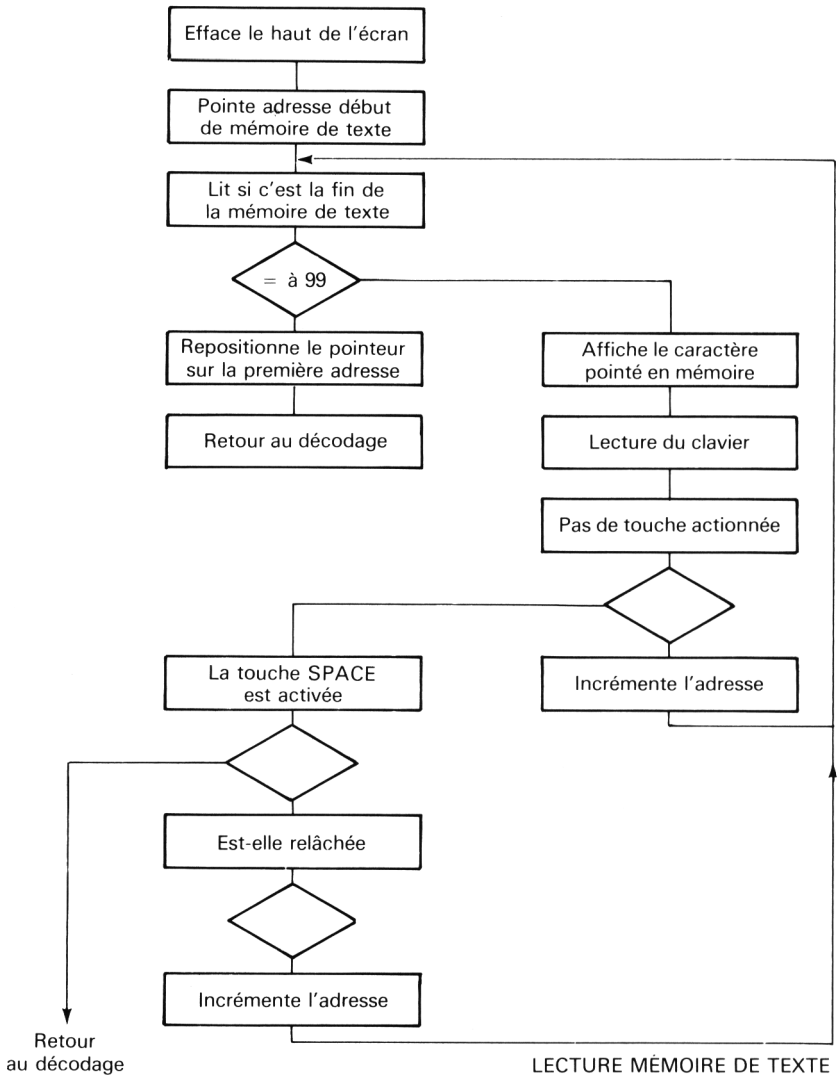


LECTURE D'UN CARACTÈRE REÇU


Lecture du registre caractère
Additionne l'offset chiffre ou lettre
Pointe la table de transcodage
Récupère le code du caractère à afficher
Teste si préfixe bascule lettre
= FF
Modifie « l'offset » table en mode lettres
Retour à la lecture du VIA après détente bit STOP
Teste si préfixe bascule chiffre
= FE
Incrémente l'adresse de la mémoire de texte
Modifie « l'offset » table en mode chiffres
Teste si mémoire encore disponible
Retour à la lecture du VIA après attent bit STOP
réinitialise au début de mémoire texte, le pointe
Stocke le caractère à l'adresse pointée
Affiche le caractère décodé
Attente à un niveau UN et retour hors attente START



LECTURE DE L'OPTION CHOISIE


Efface le haut de l'écran
Pointe adresse début de mémoire de texte
Lit si c'est la fin de la mémoire de texte
= à 99
Repositionne le pointeur sur la première adresse
Retour au décodage
Affiche le caractère pointé en mémoire
Lecture du clavier
Pas de touche actionnée
La touche SPACE est activée
Incrémente l'adresse
Est-elle relâchée
Incrémente l'adresse
Retour au décodage
LECTURE MÉMOIRE DE TEXTE

COMMENTAIRES SUR LE LISTING BASIC

Ligne 10			: abaisse le plafond de la mémoire utilisable sous BASIC
20	~	42	: lecture et implantation du langage machine rangé dans les DATA qui suivent
1000			: si on supprime REM, on exécute une boucle qui efface les « U » contenus dans la mémoire et qu'on voit apparaître lors de la lecture de la mémoire texte reçu
1003	~	1020	: initialisations diverses
1105	~	1107	: si on répond 0, pas d'émission du retour chariot automatique
1110	~	1121	: choix et initialisation de la vitesse
1125			: shift
1130	~	1150	: menu et aiguillage
1300			: 27 lignes écran-coupe générateur sonore
1302			: ouvre le relais E/R
1305			: efface CAPS
1310	~	1345	: présente les options réception
1405			: crée la fenêtre affichage 10 lignes
1420			: accès à routine de décodage
1600			: aiguillage pour réinitialisation des paramètres
2000	~	2052	: présente les différentes options émission
2056	~	2060	: aiguillage correspondant au choix
2100	~	2110	: écrit sur la ligne supérieure de l'écran
2150	~	2190	: calcule et initialise les valeurs à envoyer au générateur sonore pour MARK et SPACE
2200	~	2205	: génère le MARK et colle relais E/R
2235	~	2240	: pour le mode 0 (composition texte clavier
2247	~	2248	: pour l'option 5. Transmet à la routine machine, par DOKE l'adresse de début du texte mémorisé
2250	~	2250	: calcule pour les autres options l'adresse de début du message mémorisé correspondant
2257	~	2620	: vers les routines d'émission
3000	~	3300	: initialise et calcule la valeur de la tonalité de commande reçue, en mode automatique
3503			: ces octets sont ajoutés lors du choix de l'option « mode automatique »
3505			: donne les tolérances sur la fréquence de commande (ici 2500 $\pm$ 10 Hz)
3510			: attente réception de la bonne tonalité
3515	~	3540	: si reçue émet le texte mémorisé
3545			: temporisation de fin de message, avant de couper le MARK
3600			: coupe l'émission
3602	~	3608	: attente environ 15 secondes pour une éventuelle tonalité de commande
3610	~	3635	: si unetonalité est reçue, vérifie que c'est bien la bonne
3650			: si oui, on décode ce qui suit
4000	~	4080	: produit pendant 5 s une tonalité d'ouverture pour déclencher, en mode AUTO, l'ORIC du correspondant (option T). La valeur de la fréquence est au dénominateur ligne 4030
6000	~	6130	: composition des textes des messages mémorisés (voir remarque qui suit)

6200 ~ 6350 : implante dans les zones mémoires réservées # 7000 à # 8000 les divers messages et le texte mémorisé
7000 ~ 7100 : routine d'implantation du texte mémorisé
10000 ~ 11111 : DATA contenant le texte mémorisé. Doit se finir par @ puis !

REMARQUES IMPORTANTES

LES TEXTES MÉMORISÉS

A partir de la ligne 6000, on trouve les messages mémorisés. Vous les composerez à votre gré en respectant impérativement la longueur maximale de 255 caractères pour l'ensemble du message, y compris le Retour chariot du début et le code « ↑ » de la fin.

Nous avons fractionné les messages à cause de la limitation du nombre de caractères sur une même ligne dans le BASIC ORIC.

Ils sont assemblés par corcaténation de chaînes dans les 4 messages mémorisés possibles notés ME$ (1) à ME$ (4).

Noter que chaque message mémorisé commence par CHR$ (13) provoquant l'émission d'un retour chariot et se termine par le caractère « ↑ » rendant la main au clavier.

NOTES SUR LE LISTING

— Le signe « ↑ » (shift 6) apparaît sur le listing comme un accent circonflexe « ^ », à cause de l'imprimante... Chaque fois que vous voyez ce signe il faut donc taper shift 6.

— Ligne 2053, derrière PRINT TAB (30) ; il y a 17 blancs (pour effacer la phrase « VOTRE CHOIX ! »).

L'UTILISATION DE LA MÉMOIRE

7000 74FF : reçoit le texte mémorisé, destiné à être émis en mode AUTO. Ce texte est lu à partir des DATA de fin de programme
7500 ~ 78FF : les 4 messages mémorisés
7900 ~ 7FFF : libre. Pourrait être utilisée pour d'autres messages mémorisés
8000 ~ 8FFF : mémoire contenant le texte reçu en cours de décodage
9000 ~ 9061 : paramètres et routine génération tonalités MARK/SPACE
9160 ~ 9208 : sous-programme de saisie des caractères au clavier (pour le mode 0)
9250 ~ 9290 : mesure de la fréquence du signal de commande pour le mode déclenchement automatique
9410 ~ 94F2 : partie décodage (RÉCEPTION)
9550 ~ 95B9 : mémorisation du texte décodé (4 derniers K octets...)
9610 ~ 964B : traitement des messages mémorisés (pour leur émission
9120 ~ 915A : table de transcodage ÉMISSION
9500 ~ 953F : table de transcodage RÉCEPTION.

Variables « page ZÉRO » utilisées pendant le décodage :

39 : compteur de moments
3A : mémoire du caractère
3B : offset chiffre/lettre pour table de transcodage
3D, 3E : pointeur mémoire texte reçu
BF, 40 : pointeur début mémoire de stockage du message reçu
45 : type d'émission (avec ou sans paramètres mémorisés)

CHOIX DU DÉMODULATEUR RTTY

Le démodulateur est le circuit qui transforme les signaux BF issus du récepteur en signaux « logiques ». L'état de la sortie du démodulateur est fonction du MARK ou du SPACE.

Plusieurs schémas de démodulateurs existent dans la presse spécialisée, systèmes à filtres self/capa, systèmes à filtres actifs, systèmes à P.L.L.

Nous avons choisi ce dernier, en raison de sa simplicité de mise en œuvre. Utilisé en VHF ou en décamétrique (mais dans ce cas, sur un récepteur possédant de bons filtres) il s'avère très fiable. Nous présentons donc son schéma dans cet ouvrage.

Linterruption shift normal/shift inverse sera avec position libre au centre, à moins de ne prévoir la possibilité de déconnecter facilement la liaison sortie-démodulateur-ORIC. En effet, lorsqu'aucune réception n'a lieu (pas de tonalité) l'ORIC peut rester en attente indéfiniement et le choix d'une option au clavier est impossible. Pour sortir de cette situation il suffit de l'isoler momentanément du démodulateur.

La sortie du démodulateur (à travers un interrupteur ou non, voir ci-dessus...) ira se connecter à la prise imprimante de l'ORIC aux points :

1 pour le point chaud
2 pour la masse.

L'alimentation du démodulateur pourra se faire à partir du + 5 V de l'ORIC, disponible sur le point 33 (masse en 34) du connecteur d'extensions.

Voici donc le schéma du démodulateur à P.L.L., utilisant un circuit EXAR XR2211.

MODE D'EMPLOI DU PROGRAMME

Après branchement du démodulateur et chargement du programme, on le lance par RUN. Ceci a pour effet d'accéder aux diverses initialisations :

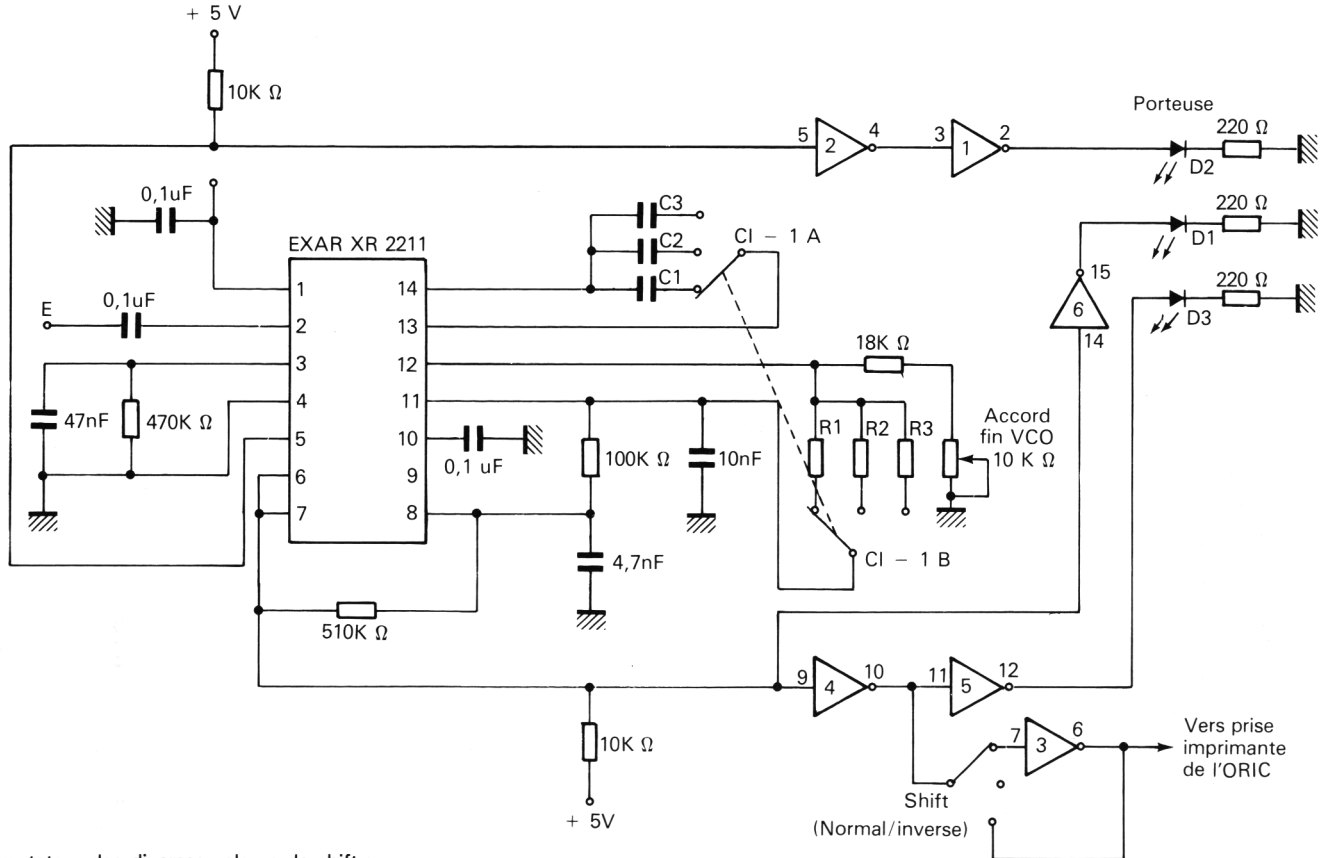
Retour-chariot : choisissez le nombre de colonnes pour son émission automatique ou tapez 0 si vous ne voulez pas émettre de R.C. automatique.

Vitesse (E/R) : vous avez le choix entre 2 vitesse 45.45 et 50 bds.

Shift (Emission seule) : choisissez votre shift normal ou inverse (dans ce cas tapez — devant) en vous rappelant les 3 valeurs les plus utilisées : 170, 425, 850 Hz.

La machine est prête alors à émettre ou recevoir.

Si vous tapez R, vous voyez apparaître alors le « Menu » avec les options en réception



Commutateur des diverses valeurs de shift :

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

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

La patte 1 du 4049 au + 5V.

— Le commutateur CI 1A / CI 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.

Caractéristiques des composants

dans la moitié inférieure de l'écran, le texte décodé apparaissant lui, sur la partie supérieure. Peu de commentaires sinon que, en affichage mémoire (M), le texte mémorisé défile très rapidement et se fige si vous maintenez la barre d'espace.

Si vous tapez E, vous accédez à l'émission.

Le mode 0 permet d'émettre un texte à partir du clavier. Vous pouvez passez instantanément du mode clavier au mode mémorisé en faisant shift 6 (↑) et en tapant l'option choisie, par exemple 3, pour les conditions de travail. A la fin du message mémorisé, la main vous sera rendue au clavier. Le passage en réception se fait par @ (shift 2).

Par le choix judicieux des changements d'options et des messages que vous aurez prédéfinis, vous pourrez ainsi passer du mode clavier au mode mémoire et vous évitez des phrases inutiles (QTH, conditions de trafic etc...).

L'option 5 provoque l'émission du texte mémorisé dans les lignes DATA commençant en 10000.

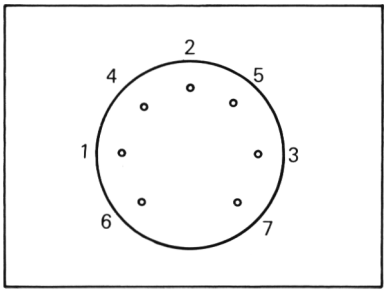
Un dernier mot : à la ligne 2200 vous avez :

SOUND 1, H, 5

en modifiant le 5 vous modifiez en rapport le niveau (volume) du son émis par ORIC. Cela pourra vous aider pour coupler ORIC à votre TX.

COUPLAGE ORIC-TX. TÉLÉCOMMANDE E/R

Se procurer une prise 7 broches 270° (DIN) et la brancher sur la sortie cassette de l'ORIC après l'avoir câblée comme suit :



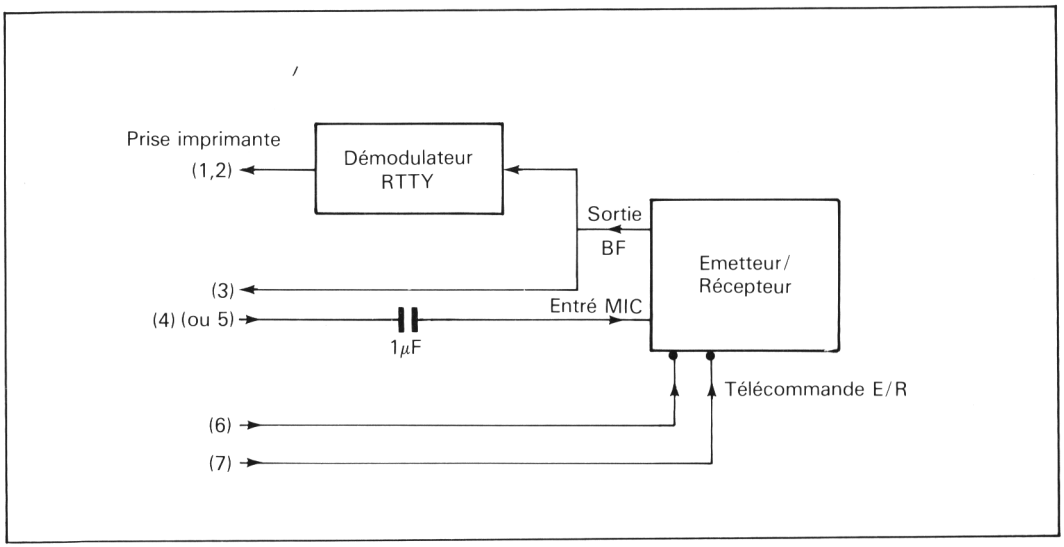
Point 1 non utilisé.

Point 3 couplé à la sortie BF du récepteur.

Point 4 ou 5 et point 2 : un blindé, âme reliée **à travers une capa** miniature au tantale de 1 μF au point 4 (ou 5) et tresse à la masse (point 2). De l'autre côté, le blindé ira à la prise micro du TX.

Points 6 et 7 deux fils qui seront reliés à la télécommande émission-réception de votre station, si vous souhaitez que le programme commande automatiquement le passage en émission-réception. (Obligatoire en mode automatique.)

SYNOPTIQUE DE L'INSTALLATION RTTY



Les liaisons entre ORIC et la station doivent être réalisées en câble blindé. La masse est sur 2 de la prise cassette de l'ORIC. Les points 3, 4 (ou 5), 6 et 7 sont ceux de cette même prise DIN.

La prise imprimante et la sortie démodulateur RTTY sont reliées par un blindé. La masse du blindé en 2 de la prise imprimante, l'âme en 1. Ces deux points sont ceux qui sont les plus proches de la prise DIN « CASSETTE ».

La liaison Sortie BF récepteur vers 3 de DIN cassette peut-être omise si on ne veut pas fonctionner en mode « automatique ». La télécommande pourra aussi être omise.

RTTY PARTIE BASIC

```
1 REM    ***** RTTY *****
2 REM    * @ D.BONOMO    *
3 REM    * & E.DUTERTRE *
4 REM    * F1EZH  F6GKQ *
5 REM    *  01-11-1983  *
6 REM    *     V.02     *
7 REM    *  O R I C - 1 *
8 REM    ****************
9 REM
10 HIMEM#7000
12 CLS:PRINT:PRINT:PRINT"PATIENTEZ QUELQUES INSTANTS"
15 DIMME$(4)
20 FORA=#9120TO#915A:READD:POKEA,D:NEXT
25 FORA=#9000TO#9061:READD:POKEA,D:NEXT
30 FORA=#9410TO#95BA:READD:POKEA,D:NEXT
35 FORA=#9160TO#920F:READD:POKEA,D:NEXT
40 FORA=#9610TO#964B:READD:POKEA,D:NEXT
42 FORA=#9250TO#9290:READD:POKEA,D:NEXT
49 REM--- TRANSCODAGE EMISSION ---
50 DATA#04,#00,#05,#00,#00,#1A,#00,#05,#0F,#12,#09,#11,#0C,#03,#1C,
#1D,#16
52 DATA#17,#13,#01,#0A,#10,#15,#07,#06
54 DATA#18,#0E,#00,#00,#1E,#00,#19,#00,#03,#19,#0E,#09,#01,#0D,#1A,
#14,#06
56 DATA#0B,#0F,#12,#1C,#0C,#18,#16,#17,#0A,#05,#10,#07,#1E,#13,#1D,
#15,#11
58 DATA#00,#05,#2B,#31,#14,#C5,#00,#00,#00,#00,#00,#00,#00,#00,#00,
#00
59 REM--- GENERATION TONALITES ---
60 DATA#98,#48,#78,#AE,#02,#90,#A9,#00,#20,#35,#F5
61 DATA#20,#55,#90,#A9,#05,#8D,#01,#90
62 DATA#EA,#6E,#00,#90,#B0,#1F,#AE,#02,#90,#A9,#00,#20,#35,#F5,#20,
#55,#90
64 DATA#CE,#01,#90,#D0,#EB,#AE,#03,#90,#A9,#00,#20,#35,#F5,#20,#51,
#90,#58
66 DATA#68,#A8,#60,#AE,#03,#90,#4C,#2C,#90
68 DATA#EA,#EA,#EA,#A0,#1E,#D0,#03,#A0,#14,#EA,#A2,#C7,#CA,#D0,#FD
70 DATA#88,#D0,#F8,#EA,#60
```

```
108 REM---- PARTIE RECEPTION ----
110 DATA#A9,#00,#85,#3B,#A9,#E7,#8D,#02,#03,#EA,#EA,#EA,#EA,#EA
120 DATA#58,#AD,#08,#02,#C9,#38,#F0,#05,#85,#35,#4C,#B0,#94
130 DATA#78,#AD,#00,#03,#29,#10,#D0,#EB,#A9,#05,#85,#39
140 DATA#20,#90,#94,#A9,#00,#85,#3A,#20,#94,#94,#A9,#10,#2D,#00,#03
150 DATA#05,#3A,#85,#3A,#C6,#39,#F0,#08,#46,#3A
160 DATA#4C,#3E,#94,#EA,#EA,#EA
170 DATA#18,#A5,#3A,#65,#3B,#85,#3A,#A6,#3A,#BD,#00,#95
175 DATA#EA,#EA,#EA,#EA,#EA
180 DATA#C9,#FF,#D0,#08,#A9,#20,#85,#3B,#4C,#86,#94,#EA
190 DATA#C9,#FE,#D0,#07,#A9,#00,#85,#3B,#4C,#86,#94,#20,#50,#95
200 DATA#AA,#20,#3F,#F7,#EA
210 DATA#AD,#00,#03,#29,#10,#F0,#F9,#4C,#1E,#94
220 DATA#A0,#0A,#D0,#03,#A0,#14,#EA,#A2,#C7,#CA
230 DATA#D0,#FD,#88,#D0,#F8,#EA,#60
235 DATA#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA
237 REM---- OPTIONS CLAVIER ----
240 DATA#A5,#35,#C9,#BA,#D0,#07,#A9,#20,#85,#3B,#4C,#2B,#94
245 DATA#C9,#8F,#D0,#07,#A9,#00,#85,#3B,#4C,#2B,#94
250 DATA#C9,#B6,#D0,#08,#A2,#0C,#20,#3F,#F7,#4C,#2B,#94
255 DATA#C9,#82,#D0,#06,#20,#79,#95,#4C,#2B,#94
260 DATA#C9,#9E,#D0,#05,#A9,#01,#85,#45,#60
265 DATA#C9,#89,#D0,#05,#A9,#00,#85,#45,#60,#4C,#2B,#94
270 DATA#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA
349 REM--- TRANSCODAGE RECEPTION ---
350 DATA#20,#45,#0A,#41,#20,#53,#49,#55,#0D,#44,#52,#4A,#4E,#46,#43
,#4B,#54
360 DATA#5A,#4C,#57,#48,#59,#50,#51,#4F,#42,#47,#FF,#4D,#58,#56,#FE
370 DATA#21,#33,#0A,#2D,#20,#22,#38,#37,#0D,#2A,#34,#0A,#2C,#45,#3A
,#28,#35
380 DATA#2B,#29,#32,#48,#36,#30,#31,#39,#3F,#25,#FF,#2E,#2F,#3D,#FE
385 DATA#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA
,#EA
390 REM---- MEMOR TEXTE ----
400 DATA#85,#3A,#18,#A9,#01,#65,#3D,#85,#3D,#A9,#00,#65,#3E,#85,#3E
405 DATA#A2,#01,#A1,#3C,#C9,#99,#D0,#08,#A9,#00,#85,#3D
410 DATA#A9,#80,#85,#3E,#A5,#3A,#81,#3C,#60
415 DATA#EA,#EA,#EA,#EA,#EA,#AD,#08,#02,#C9,#38,#D0,#F9
420 DATA#A2,#0C,#20,#3F,#F7,#18,#A9,#01,#65,#3F,#85,#3F,#A9,#00
425 DATA#65,#40,#85,#40,#A2,#01,#A1,#3E,#C9,#99,#D0,#09
430 DATA#A9,#00,#85,#3F,#A9,#80,#85,#40,#60
435 DATA#AA,#20,#3F,#F7,#AD,#08,#02,#C9,#38,#F0,#D7
440 DATA#C9,#84,#F0,#F5,#A2,#0C,#20,#3F,#F7,#4C,#9A,#95,#EA
```

```
499 REM---- SAISIE MACHINE -----
500 DATA#A9,#00,#A8,#20,#F8,#C5,#8D,#06,#90,#C8,#99,#00,#93
505 DATA#EA,#EA,#EA,#EA,#EA,#EA
510 DATA#C9,#40,#F0,#04,#C9,#5E,#D0,#07,#8D,#00,#91,#EA,#EA,#EA,#60
515 DATA#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA
520 DATA#AA,#20,#3F,#F7,#C9,#0D,#D0,#06,#20,#F0,#91,#4C,#60,#91
525 DATA#AA,#20,#AD,#91,#CE,#02,#91,#D0,#03,#20,#F0,#91,#4C,#60,#91
527 DATA#C9,#20,#F0,#31
530 DATA#C9,#40,#B0,#1B,#AD,#03,#91,#D0,#28,#A9,#01,#8D,#03,#91
535 DATA#A9,#1B,#8D,#00,#90,#20,#10,#90,#4C,#E2,#91,#EA,#EA,#EA,#EA
,#EA
540 DATA#EA,#AD,#03,#91,#F0,#0E,#A9,#00,#8D,#03,#91,#A9,#1F,#8D,#00
,#90
545 DATA#20,#10,#90,#AE,#06,#90,#BD,#00,#91,#8D,#00,#90,#20,#10,#90
,#60,#EA
550 DATA#A2,#0A,#20,#3F,#F7,#AD,#01,#91,#8D,#02,#91,#A9,#08,#8D,#00
,#90
555 DATA#20,#10,#90,#A9,#02,#8D,#00,#90,#20,#10,#90,#60,#EA,#EA,#EA
,#EA
599 REM---- LEC. MES. MEMO.----
600 DATA#AC,#07,#90,#B9,#00,#75,#8D,#06,#90,#C9,#40,#F0,#04
605 DATA#C9,#5E,#D0,#07,#8D,#00,#91,#EA,#EA,#EA,#60
610 DATA#AA,#20,#3F,#F7,#C9,#0D,#D0,#06,#20,#F0,#91
615 DATA#4C,#41,#96,#20,#AD,#91,#CE,#02,#91,#D0,#03
620 DATA#20,#F0,#91,#EE,#07,#90,#D0,#03,#EE,#15,#96,#4C,#10,#96
699 REM---- MESURE FREQUENCE ----
700 DATA#78,#AE,#00,#04,#8E,#08,#03,#AE,#01,#04,#AD,#00,#03
705 DATA#A9,#10,#2C,#0D,#03,#F0,#FB,#8E,#09,#03,#AE,#00,#03
710 DATA#A0,#00,#AA,#3A,#2C,#0D,#03,#F0,#FB,#AE,#00,#03,#C8,#AA
715 DATA#A9,#20,#2C,#0D,#03,#F0,#EE,#AE,#02,#04,#F0,#0B,#98
720 DATA#9D,#10,#04,#CA,#8E,#02,#04,#4C,#51,#92,#58,#60
999 REM-----------------------------------
1000 REM FORA=#8000TO#8FFE:POKEA,32:NEXT
1002 INK7:PAPER0
1003 POKE#45,1
1005 POKE#3D,00:POKE#3E,128:POKE#3F,00:POKE#40,128:POKE#8FFF,153
1008 GOSUB6000
1010 PRINTCHR$(6):PRINTCHR$(12)
1015 POKE#9100,64:POKE#9103,0
1020 L=1:C=0
1050 REM---- PRESENTATION ----
1060 PRINTTAB(22);CHR$(4);CHR$(27);"J----RTTY----";CHR$(4)
1065 PRINT
1070 PRINTTAB(22);CHR$(27);"LF1EZH--F6GKQ"
1080 FORA=#BBA4TO#BBA7:POKEA,32:NEXT
1100 REM---- REINITIALISATION ----
1105 PRINT:PRINT:INPUT"NB. DE COLONNES POUR CR AUTO :";K
```

```
1106 POKE#9101,K:POKE#9102,K
1107 IFK=0THENNRC=1ELSENRC=0
1110 PRINT:PRINT:INPUT"VITESSE (45 OU 50 BAUDS):";V
1115 IFV<>50THEN1119
1116 REM---- 50 BDS ----
1117 POKE#9491,10:POKE#9495,20:POKE#9056,20
1118 POKE#9498,199:POKE#9059,199:POKE#9052,30:GOTO1125
1119 REM---- 45 BDS ----
1120 POKE#9491,11:POKE#9495,22:POKE#9056,22
1121 POKE#9498,199:POKE#9059,199:POKE#9052,33
1125 PRINT:PRINT:INPUT"SHIFT (+/-HERTZ):";S
1130 PRINT:PRINT:PRINT:PRINT" E - Pour Emettre ":PRINT" R - Pour Re
cevoir"
1132 PRINT" A - Pour Automatique"
1133 PRINT" T - Pour Tonalite de commande"
1135 POKE#26A,43
1140 ETA$:IFA$="E"THEN2000
1145 IFA$="A"THEN3500
1147 IFA$="T"THENGOTO4000
1150 IFA$<>"R"THEN1140
1299 REM---- S/P DE RECEPTION ----
1300 POKE#26F,27: CLS:SOUND1,49,0
1302 POKE#302,231
1305 FORA=#BB8BTO#BB9A:POKEA,32:NEXT
1310 POKE#268,15
1315 POKE#269,2
1317 PRINT
1320 PRINT"   C - FORCAGE CHIFFRES"
1325 PRINT"   L - FORCAGE LETTRES"
1330 PRINT"   S - EFFACAGE ECRAN"
1335 PRINT"   M - AFFICHE MEMOIRE"
1340 PRINT"   E - EMISSION ";S;" HZ  ";V;" BDS"
1345 PRINT"   T - EMISSION AVEC MODIF. PARAMETRES"
1405 POKE#26F,10
1415 CLS
1420 CALL#9410
1500 POKE#26F,27
1505 A$=KEY$
1510 CLS
1600 IFPEEK(#45)=0THEN1010ELSE2000
1999 REM---- S/P D'EMISSION ----
2000 CLS
2002 POKE#26F,10
2005 POKE#268,14:POKE#269,2:PRINT
2007 PRINT"@ POUR PASSER EN RECEPTION"
2008 PRINT:PRINT"^ POUR CHANGER OPTION EMISSION"
2009 PRINT
```

```
2010 PRINT"0 Emission a Partir du clavier"
2020 PRINT"1 Emission du messa9e d'aPPel"
2025 PRINT"2 Emission du messa9e de test"
2030 PRINT"3 Emission du messa9e memorise"
2035 PRINT"4 Emission d'une li9ne de RY"
2040 PRINT"5 Emission de la memoire de texte"
2050 PRINT: PRINTTAB(30);"-> VOTRE CHOIX ? ";
2052 GETRE$
2053 POKE#268,24:POKE#269,18:PRINT:PRINT:PRINTTAB(30);"
     "
2056 IF(RE$<>"@")OR(RE$<>"^")THENRE=VAL(RE$)
2057 IFRE$="@"THEN1300
2058 IFRE$="^"THEN2330
2060 CLS:GOSUB2100:GOTO2230
2099 REM----- PREPARE EMISSION -----
2100 A$=" F1EZH - F6GK0 "
2110 FORA=1TOLEN(A$):POKE#BB8A+A,ASC(MID$(A$,A,1)):NEXT
2150 IFS>0THENQ=1275:W=Q+S
2160 IFS<0THENW=1275:Q=W-S
2180 H=INT(62500/Q):B=INT(62500/W)
2190 POKE#9003,H:POKE#9002,B
2200 SOUND1,H,5
2205 POKE#302,167
2210 RETURN
2230 IFRE<>0THEN2245
2235 CALL#9160
2240 GOTO2300
2245 IFRE<>5THEN2250
2247 DOKE#9614,28672
2248 GOTO2257
2250 RE=RE-1
2255 DOKE#9614,(29952+(256*RE))
2257 POKE#9007,0
2260 CALL#9610
2300 IFPEEK(#9100)=#40THEN1300
2305 IFPEEK(#9100)=#5ETHEN2330
2330 POKE#268,24:POKE#269,18:PRINT:GOTO2050
2620 GOTO2230
2999 REM----- AUTOMATIQUE -----
3000 POKE#400,232:POKE#401,253:POKE#402,10
3100 CALL#9250
3200 I=0:FORA=0TO9:I=I+(PEEK(#041A-A)):NEXT
3210 FCE=INT(I/65E-2)
3220 IF(FCE<SUP)AND(FCE>INF)THENFCE=1ELSEFCE=0
3300 RETURN
```

```
3500 REM------ E/R AUTO ------
3502 CLS:PRINT:PRINT:PRINT"ATTENTE DECLENCHEMENT..."
3503 DOKE#9462,#21C9:DOKE#9464,#01D0:DOKE#94A1,#6058
3505 INF=2490:SUP=2510
3510 GOSUB3000
3512 K$=KEY$:IFK$<>""THEN1130
3513 IFFCE=0THEN3510
3515 CLS:GOSUB2100:WAIT500
3520 DOKE#9614,28672:POKE#9007,0
3540 CALL#9610
3545 WAIT1000
3600 POKE#302,231:SOUND1,49,0:CLS
3602 A=PEEK(#300)
3604 FORN=1TO1500
3606 IF(PEEK(#30D)=50)THEN3610
3607 NEXTN
3608 GOTO3502
3610 INF=2490:SUP=2510
3620 FORN=1TO5:GOSUB3000
3630 IFFCE=1THEN3650
3635 NEXTN
3640 GOTO3502
3650 CALL#9410
3660 GOTO3502
3699 REM----- TONALITE OUVERTURE -----
4000 PRINT:PRINT"POUR GENERER LA TONALITE DE COMMANDE     APPUYER SU
R UNE TOUCHE"
4010 GETK$
4020 POKE#302,167
4030 T=INT(62500/2500)
4040 SOUND1,T,7
4050 WAIT500
4060 SOUND1,49,0
4070 POKE#302,231
4080 CLS:GOTO1130
5999 REM------ MESSAGES MEMORISES -----
6000 RY$="RYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRY"
6010 AP$="    CQ CQ CQ DE F6GKQ F6GKQ F6GKQ / 91    "
6020 TE$="  TEST TEST DE F6GKQ / 91 TEST TEST      "
6030 CH$="    0123456789 ' * - , . ( ) + / = %  :    "
6040 LE$="         QWERTYUIOPASDFGHJKLZXCVBNM         "
6050 CO$=" MICRO-COMPUTER ORIC-1.SOFT F1EZH-F6GKQ "
6060 ST$="TX MULTI 750E,DECODEUR PLL,ANT 14EL/22M "
6100 ME$(1)=CHR$(13)+RY$+AP$+AP$+AP$+RY$+"^"
6110 ME$(2)=CHR$(13)+RY$+TE$+CH$+LE$+RY$+"^"
6120 ME$(3)=CHR$(13)+CO$+ST$+"^"
```

```
6130 ME$(4)=CHR$(13)+RY$+"^"
6200 FORI=0TO3
6210 A=#7500+(I*256)
6215 POKEA,13
6220 GOSUB6300:NEXT
6225 GOSUB7000
6230 RETURN
6300 J=1
6310 REPEAT
6320 C$=MID$(ME$(I+1),J,1)
6330 POKE(A+J),ASC(C$)
6340 J=J+1
6350 UNTILC$="^"
6360 RETURN
6999 REM------IMPLANTE TEXTE MEMO------
7000 A=#7000
7005 POKE#7000,13
7010 A=A+1
7015 READD$
7020 FORI=1TOLEN(D$)
7025 C$=MID$(D$,I,1)
7030 POKEA,ASC(C$):A=A+1:NEXT
7050 READD$:IFD$<>"!"THEN7020
7100 RETURN
9999 REM--- MESSAGE AUTOMATIQUE ---
10000 DATA"RYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRY    F6GKQ / 91 COR
BEIL"
10005 DATA"                        EMISSION AUTOMATIQUE PAR MICRO ORIC-
1. "
10010 DATA"   LOGICIEL DEVELOPPE PAR F1EZH ET F6GKQ   "
10015 DATA"LA STATION EST UN MULTI-750E 10W HF     ANTENNE 14 EL A
22M...@"
11111 DATA"!"
```

PROGRAMME E/R RTTY

PARTIE ASSEMBLEUR

SOUS-PROGRAMME DE GÉNÉRATION DES TONALITÉS MARK-SPACE

```
?
02
9010 98      TYA
9011 48      PHA
9012 78      SEI
9013 AE0290  LDX #9002
9016 A900    LDA %#00
9018 2035F5  JSR #F535
901B 205590  JSR #9055
901E A905    LDA %#05
9020 8D0190  STA #9001
9023 EA      NOP
9024 6E0090  ROR #9000
9027 B01F    BCS #9048
9029 AE0290  LDX #9002
902C A900    LDA %#00
902E 2035F5  JSR #F535
9031 205590  JSR #9055
9034 CE0190  DEC #9001
9037 D0EB    BNE #9024
9039 AE0390  LDX #9003
903C A900    LDA %#00
903E 2035F5  JSR #F535
9041 205190  JSR #9051
9044 58      CLI
9045 68      PLA
9046 A8      TAY
9047 60      RTS
9048 AE0390  LDX #9003
904B 4C2C90  JMP #902C
904E EA      NOP
904F EA      NOP
9050 EA      NOP
```

Sauvegarde du contenu de l'accumulateur

Inhibition des interruptions

Initialise générateur sonore pour fréquence SPACE

Temporisation
Initialise le compteur de moments
et le mémorise

Le code BAUDOT du caractère à émettre est décalé à travers le bit CARRY. Si c'est un 1 on génère un MARK sinon SPACE

Génération du SPACE

Moment suivant ; si dernier, génère MARK bit de STOP

Emet un SPACE

Tempo pour bit de STOP
Autorisation des interruptions

Restitution du contexte et retour

Initialise avec fréquence du MARK, si moment = 0

```
9051 A021    LDY %#21
9053 D003    BNE #9058
9055 A016    LDY %#16
9057 EA      NOP
9058 A2C7    LDX %#C7
905A CA      DEX
905B D0FD    BNE #905A
905D 88      DEY
905E D0F8    BNE #9058
9060 EA      NOP
9061 60      RTS
```

Tempo longue

Tempo courte

sous-programme de DELAI

SOUS-PROGRAMME DE SAISIE D'UN CARACTÈRE AU CLAVIER

```
0B
9160 A900    LDA %#00
9162 A8      TAY
9163 20F8C5  JSR #C5F8
9166 8D0690  STA #9006
9169 C8      INY
916A 990093  STA #9300,Y
916D EA      NOP
916E EA      NOP
916F EA      NOP
9170 EA      NOP
9171 EA      NOP
9172 EA      NOP
9173 C940    CMP %#40
9175 F004    BEQ #917B
9177 C95E    CMP %#5E
9179 D007    BNE #9182
917B 8D0091  STA #9100
917E EA      NOP
917F EA      NOP
9180 EA      NOP
9181 60      RTS
9182 EA      NOP
9183 EA      NOP
9184 EA      NOP
```

(Pas utilisé)

Saisie du caractère par la routine en ROM
Mémorisation du caractère

(Pas utilisé)

Teste si c'est @ pour passage en réception (SHIFT 2)

Teste si c'est ↑ pour changement OPTION (SHIFT 6)

Mémorise l'un de ces deux choix avant

le retour au BASIC

Début du traitement de la touche saisie

```
9185 EA      NOP
9186 EA      NOP
9187 EA      NOP
9188 EA      NOP
9189 EA      NOP
918A EA      NOP
918B EA      NOP
918C EA      NOP
918D EA      NOP
918E EA      NOP
918F EA      NOP
9190 AA      TAX          Affichage du caractère correspondant par la
9191 203FF7  JSR #F73F    routine de la ROM.
9194 C90D    CMP %#0D     Teste si c'est un RETURN
9196 D006    BNE #919E
9198 20F091  JSR #91F0
919B 4C6091  JMP #9160    Vers le S/P traitement RETURN
919E AA      TAX          Retour au début de saisie
919F 20AD91  JSR #91AD    Sauvegarde avant traitement et
91A2 CE0291  DEC #9102    émission du caractère
91A5 D003    BNE #91AA    tenue à jour du compteur de colonnes
91A7 20F091  JSR #91F0    Si nul on
91AA 4C6091  JMP #9160    émet retour chariot et avance papier puis
91AD C920    CMP %#20     retour à saisie
91AF F031    BEQ #91E2    Teste si espace
91B1 C940    CMP %#40
91B3 B01B    BCS #91D0    Test chiffre/lettre
91B5 AD0391  LDA #9103
91B8 D028    BNE #91E2    En # 9103 on a 1 mode Chiffre 0 si mode Lettre
91BA A901    LDA %#01     déjà en mode chiffre
91BC 8D0391  STA #9103    Charge la bascule Chiffre/Lettre
91BF A91B    LDA %#1B     Charge le préfixe chiffre (code BAUDOT)
91C1 8D0090  STA #9000    et saute au
91C4 201090  JSR #9010    sous-programme émission des tonalités
91C7 4CE291  JMP #91E2    Saute à préparation émission
91CA EA      NOP
91CB EA      NOP
91CC EA      NOP
91CD EA      NOP
91CE EA      NOP
91CF EA      NOP
91D0 AD0391  LDA #9103    Teste si déjà en mode LETTRE
91D3 F00E    BEQ #91E3    Si oui saute à préparation émission
91D5 A900    LDA %#00     Charge la bascule en mode LETTRE
91D7 8D0391  STA #9103
91DA A91F    LDA %#1F     Charge le préfixe LETTRE (code BAUDOT)
91DC 8D0090  STA #9000    Emet le préfixe Lettre
```

```
91DF 201090 JSR #9010
91E2 AE0690 LDX #9006        Récupère
91E5 BD0091 LDA #9100,X      Pointe dans la table de transcodage émission
91E8 8D0090 STA #9000        Le code BAUDOT est transféré pour
91EB 201090 JSR #9010        la routine de génération Tonalité
91EE 60     RTS
91EF EA     NOP
91F0 A20A   LDX %#0A         Curseur un                RETOURS CHARIOT
91F2 203FF7 JSR #F73F        Cran vers le bas (pour la visu)
91F5 AD0191 LDA #9101        Réinitialise le
91F8 8D0291 STA #9102        Compteur de colonnes
91FB A908   LDA %#08         Code BAUDOT de
91FD 8D0090 STA #9000        l'avance papier transmis à
9200 201090 JSR #9010        routine Tonalités
9203 A902   LDA %#02         Code BAUDOT du
9205 8D0090 STA #9000        Retour chariot transmis à
9208 201090 JSR #9010        routine Tonalités
```

SOUS-PROGRAMME DE « MESURE DE FRÉQUENCE »

```
01
9250 78     SEI
9251 AE0004 LDX #0400        inhibition des interruptions
9254 8E0803 STX #0308        initialise l'octer de poids
9257 AE0104 LDX #0401        faible du temporisateur 2 du VIA
925A AD0003 LDA #0300        charge X avec la valeur octet poids fort
925D A910   LDA %#10
925F 2C0D03 BIT #030D        Teste le BIT 4 du registre IFR (entrée CB1')
9262 F0FB   BEQ #925F
9264 8E0903 STX #0309        Démarre le comptage en chargeant la valeur de l'octet
9267 AE0003 LDX #0300        de poids fort du TIMER 2
926A A000   LDY %#00
926C AA     TAX
926D 8A     TXA
926E 2C0D03 BIT #030D        Attente d'une interruption due à l'entrée mesure
9271 F0FB   BEQ #926E
9273 AE0003 LDX #0300
9276 C8     INY
9277 AA     TAX              Comptage des impulsions reçues
9278 A920   LDA %#20
927A 2C0D03 BIT #030D        Teste le BITS de IFR (Timer 2 à zéro ?)
927D F0EE   BEQ #926D
927F AE0204 LDX #0402
9282 F00B   BEQ #928F        Teste si c'est la dernière mesure
9284 98     TYA
```

```
9285 9D1004 STA #0410,X
9288 CA     DEX
9289 8E0204 STX #0402
928C 4C5192 JMP #9251
928F 58     CLI
9290 60     RTS
```

Mémorise le résultat de la mesure
décrémente le compteur de
mesures
Recommence...
Retour après la dernière mesure, en ré-autorisant les interruptions

SOUS-PROGRAMME PRINCIPAL DE LA PARTIE RÉCEPTION

```
03
9410 A900    LDA %#00
9412 853B    STA #3B
9414 A9E7    LDA %#E7
9416 8D0203  STA #0302
9419 EA      NOP
941A EA      NOP
941B EA      NOP
941C EA      NOP
941D EA      NOP
941E 58      CLI
941F AD0802  LDA #0208
9422 C938    CMP %#38
9424 F005    BEQ #942B
9426 8535    STA #35
9428 4CB094  JMP #94B0
942B 78      SEI
942C AD0003  LDA #0300
942F 2910    AND %#10
9431 D0EB    BNE #941E
9433 A905    LDA %#05
9435 8539    STA #39
9437 209094  JSR #9490
943A A900    LDA %#00
943C 853A    STA #3A
943E 209494  JSR #9494
9441 A910    LDA %#10
9443 2D0003  AND #0300
9446 053A    ORA #3A
9448 853A    STA #3A
944A C639    DEC #39
944C F008    BEQ #9456
944E 463A    LSR #3A
9450 4C3E94  JMP #943E
9453 EA      NOP
9454 EA      NOP
9455 EA      NOP
9456 18      CLC
9457 A53A    LDA #3A
9459 653B    ADC #3B
945B 853A    STA #3A
```

initialisation arbitraire en mode LETTRES

Programme VIA pour PB4 en entrée (PB3 y est déjà)
profil binaire 11100111

Autorise les interruptions (pour lecture clavier pendant un instant)

Teste si une touche est actionnée sur le clavier

Mémorise le code de la touche actionnée

Inhibe les interruptions
Lecture de la
ligne PB4 VIA pour attente d'un niveau bas (START)

Initialise le compteur de moments et
mémorise
Délai 1/2 moment pour centrage sur milieu du bit de START

initialise la mémoire qui contiendra le caractère
Délai 1 moment pour centrage sur bit suivant

Lecture de PB4
Reconstitution du profil BAUDOT du caractère reçu
dans # 3A
Teste si dernier moment

Décalage du profil binaire du caractère en cours de réception
Au bit suivant...

Début de traitement du caractère reçu

En # 3B on a l'offset chiffre/lettre pour l'adressage table

```
945D A63A     LDX #3A
945F BD0095   LDA #9500,X
9462 C921     CMP %#21
9464 D001     BNE #9467
9466 EA       NOP
9467 C9FF     CMP %#FF
9469 D008     BNE #9473
946B A920     LDA %#20
946D 853B     STA #3B
946F 4C8694   JMP #9486
9472 EA       NOP
9473 C9FE     CMP %#FE
9475 D007     BNE #947E
9477 A900     LDA %#00
9479 853B     STA #3B
947B 4C8694   JMP #9486
947E 205095   JSR #9550
9481 AA       TAX
9482 203FF7   JSR #F73F
9485 EA       NOP
9486 AD0003   LDA #0300
9489 2910     AND %#10
948B F0F9     BEQ #9486
948D 4C1E94   JMP #941E
9490 A00B     LDY %#0B
9492 D003     BNE #9497
9494 A016     LDY %#16
9496 EA       NOP
9497 A2C7     LDX %#C7
9499 CA       DEX
949A D0FD     BNE #9499
949C 88       DEY
949D D0F8     BNE #9497
949F EA       NOP
94A0 60       RTS
94A1 58       CLI
94A2 60       RTS
94A3 EA       NOP
94A4 EA       NOP
94A5 EA       NOP
94A6 EA       NOP
94A7 EA       NOP
94A8 EA       NOP
```

Calcule le rang, dans la table, du caractère reçu
Pointe dans la table de transcodage
Test si c'est « ! », seulement en mode fonctionnement automatique, ces 4 octets sont replacés par des NOP, sinon

Teste si c'est le préfixe CHIFFRE

Modifie l'offset table transcodage, en conséquence

et va attendre le bit de STOP

Teste si c'est le préfixe LETTRE

Modifie l'offset table en conséquence

et va attendre le bit de STOP

Affiche le caractère lu dans la table par la routine affichage de l'ORIC

Attente du bit de STOP ou continue vers caractère suivant

Retour à scrutation clavier
initialise pour délai court (1/2 moment)

initialise pour délai long (1 moment)

Sous-programme de DELAI

Utilisé seulement en mode fonctionnement automatique, pour le passage en veille après réception d'un « ! »

```
94A9 EA      NOP
94AA EA      NOP
94AB EA      NOP
94AC EA      NOP
94AD EA      NOP
94AE EA      NOP
94AF EA      NOP
94B0 A535    LDA #35
94B2 C9BA    CMP %#BA
94B4 D007    BNE #94BD
94B6 A920    LDA %#20
94B8 853B    STA #3B
94BA 4C2B94  JMP #942B
94BD C98F    CMP %#8F
94BF D007    BNE #94C8
94C1 A900    LDA %#00
94C3 853B    STA #3B
94C5 4C2B94  JMP #942B
94C8 C9B6    CMP %#B6
94CA D008    BNE #94D4
94CC A20C    LDX %#0C
94CE 203FF7  JSR #F73F
94D1 4C2B94  JMP #942B
94D4 C982    CMP %#82
94D6 D006    BNE #94DE
94D8 207995  JSR #9579
94DB 4C2B94  JMP #942B
94DE C99E    CMP %#9E
94E0 D005    BNE #94E7
94E2 A901    LDA %#01
94E4 8545    STA #45
94E6 60      RTS
94E7 C989    CMP %#89
94E9 D005    BNE #94F0
94EB A900    LDA %#00
94ED 8545    STA #45
94EF 60      RTS
94F0 4C2B94  JMP #942B
```

RECONNAISSANCE TOUCHE ACTIONNÉE

Relecture du code de la touche actionnée
C = forçage du mode CHIFFRE

Ecrit valeur offset correspondant pour la table de transcodage

L = forçage du mode LETTRE

Ecrit valeur offset correspondant pour la table de transcodage

S = effacement de l'écran

Exécute un CTRL L (PRINT CHR$ (12))

M = lecture de la mémoire « texte reçu »

Saute à la routine d'affichage de la mémoire de texte

E = Emission sur paramètres Shift et Vitesse déjà mémorisés

Initialise la variable d'état émission

T = Emission avec des paramètres Shift et Vitesse différents

Initialise la variable d'état émission

Si toute autre touche actionnée, retour au décodage

MÉMORISATION DU TEXTE REÇU (Mémoire de texte 4095 octets)

```
0A
9550 853A    STA #3A
9552 18      CLC
9553 A901    LDA %#01
9555 653D    ADC #3D
9557 853D    STA #3D
9559 A900    LDA %#00
955B 653E    ADC #3E
955D 853E    STA #3E
955F A201    LDX %#01
9561 A13C    LDA (#3C,X)
9563 C999    CMP %#99
9565 D008    BNE #956F
9567 A900    LDA %#00
9569 853D    STA #3D
956B A980    LDA %#80
956D 853E    STA #3E
956F A53A    LDA #3A
9571 813C    STA (#3C,X)
9573 60      RTS
9574 EA      NOP
9575 EA      NOP
9576 EA      NOP
9577 EA      NOP
9578 EA      NOP
9579 AD0802  LDA #0208
957C C938    CMP %#38
957E D0F9    BNE #9579
9580 A20C    LDX %#0C
9582 203FF7  JSR #F73F
9585 18      CLC
9586 A901    LDA %#01
9588 653F    ADC #3F
958A 853F    STA #3F
958C A900    LDA %#00
958E 6540    ADC #40
9590 8540    STA #40
9592 A201    LDX %#01
9594 A13E    LDA (#3E,X)
9596 C999    CMP %#99
```

Réserve le dernier caractère reçu

Incrémente l'adresse du pointeur mémoire

Compare la dernière position du pointeur mémoire avec « la butée » indiquant fin de mémoire disponible (# 8FFF)

Si mémoire, on repositionne le pointeur en (# 8000)

Charge en mémoire le dernier caractère reçu

Attente que l'opérateur relâche la touche option M

Efface l'écran

Positionne le pointeur sur l'adresse du caractère mémorisé qu'il faut afficher

Teste si on est rendu en haut de la mémoire de texte (où l'on trouvera # 99)

```
9598 D009    BNE #95A3
959A A900    LDA %#00
959C 853F    STA #3F
959E A980    LDA %#80
95A0 8540    STA #40
95A2 60      RTS
95A3 AA      TAX
95A4 203FF7  JSR #F73F
95A7 AD0802  LDA #0208
95AA C938    CMP %#38
95AC F0D7    BEQ #9585
95AE C984    CMP %#84
95B0 F0F5    BEQ #95A7
95B2 A20C    LDX %#0C
95B4 203FF7  JSR #F73F
95B7 4C9A95  JMP #959A
```

Quand le haut de RAM texte est atteint, on remet le pointeur sur l'adresse # 8000 est on revient

au décodage

Affichage du caractère pointé en mémoire

Si pas de touche clavier actionnée, on continue l'affichage

Si touche ESPACE actionnée, on interrompt l'affichage pendant la durée de l'appui

SOUS-PROGRAMME DE TRAITEMENT DES MESSAGES MÉMORISÉS

```
9610 AC0790  LDY #9007
9613 B90075  LDA #7500,Y
9616 8D0690  STA #9006
9619 C940    CMP %#40
961B F004    BEQ #9621
961D C95E    CMP %#5E
961F D007    BNE #9628
9621 8D0091  STA #9100
9624 EA      NOP
9625 EA      NOP
9626 EA      NOP
9627 60      RTS
9628 AA      TAX
9629 203FF7  JSR #F73F
962C C90D    CMP %#0D
962E D006    BNE #9636
9630 20F091  JSR #91F0
9633 4C4196  JMP #9641
9636 20AD91  JSR #91AD
```

Donne l'adresse du premier octet du message sélectionné

Teste si le caractère est @ (retour réception) — SHIFT 2 —

Teste si le caractère est ↑ (change option) — SHIFT 6 —
Sinon, on traite
Mémorisé et retour

Affiche le caractère pris dans le message

Teste si c'est un RETURN

Si oui appel du sous-programme Retour-Chariot

```
9639 CE0291 DEC #9102
963C D003   BNE #9641
963E 20F091 JSR #91F0
9641 EE0790 INC #9007
9644 D003   BNE #9649
9646 EE1596 INC #9615
9649 4C1096 JMP #9610
```

Saut au sous-programme de traitement
Comptage Colonnes
SI dernière colonne émet un
Retour-Chariot + Avance-papier
Incrémente le compteur de caractères
et si > 255
Incrémente adresse pour prochain caractère (MSByte
Retour au début

TABLE DE TRANSCODAGE ÉMISSION

```
0B

9120 04 00 05 00 00 1A 00 05 ........
9128 0F 12 09 11 0C 03 1C 1D ........
9130 16 17 13 01 0A 10 15 07 ........
9138 06 18 0E 00 00 1E 00 19 ........
9140 00 03 19 0E 09 01 0D 1A ........
9148 14 06 0B 0F 12 1C 0C 18 ........
9150 16 17 0A 05 10 07 1E 13 ........
9158 1D 15 11
```

TABLE DE TRANSCODAGE RÉCEPTION

```
01

9500 20 45 0A 41 20 53 49 55  E.A SIU
9508 0D 44 52 4A 4E 46 43 4B .DRJNFCK
9510 54 5A 4C 57 48 59 50 51 TZLWHYPQ
9518 4F 42 47 FF 4D 58 56 FE OBG.MXV.
9520 21 33 0A 2D 20 22 38 37 !3.- "87
9528 0D 2A 34 0A 2C 45 3A 28 .*4.,E:(
9530 35 2B 29 32 48 36 30 31 5+)2H601
9538 39 3F 25 FF 2E 2F 3D FE 9?%../=.
9540 EA
```

TRANSFORMEZ VOTRE ORIC-1 EN MAGNÉTOPHONE NUMÉRIQUE

Grâce à ce programme en langage machine, votre ORIC va pouvoir enregistrer dans sa mémoire votre voix et la reproduire quand vous la désirerez. Par quel moyen ?

N'oubliez pas que l'ORIC à une oreille et une bouche via la prise Magnétophone, c'est donc par là que tout va se passer. Tout d'abord, il vous faudra enregistrer un message sur une cassette puis le faire lire par l'ordinateur. Ce message à une durée maximale variable (entre 45 secondes et 1 minute) suivant son contenu. Une fois le message en mémoire, il vous sera possible alors de le faire répéter par votre machine en lui connectant un amplificateur sur la prise « Mic » ou réenregistrer sur k7. Le signal enregistré n'étant qu'un reflet des fréquences et non des amplitudes, ne vous attendez pas à subir de la HI-FI mais malgré toutes les déformations, vous reconnaîtrez votre voix.

Attention toutefois au niveau injecté par le magnétophone dans l'ordinateur. A cause du trigger d'entrée de l'ORIC, un signal trop fort sera noyé dans le bruit, un signal trop faible sera haché.

Le premier travail à effectuer est d'entrer le programme :

Programme 1 : Il va vous permettre d'entrer le programme principal :

```
10  HIMEM # 07F0
20  L = O7FF : M = L
30  INPUT A$ : IFA$ = « FF » THEN 100
40  B$ = « # » + A$ : C = VAL(B$) : POKE L, C : L = L + 1
50  GOTO 30
100 IFM = # 07FF THEN L = # 0900 : M = L : GOTO 30
110 END.
```

ensuite faites RUN et tapez les données suivantes en les validant chacune par « RETURN ». Une fois ce travail fait, sauvegardez le tout sur k7 en faisant : CSAVE « MAGNETO », A # 07F0, E # 0938.

Pour recharger plus tard le programme, il suffira de faire :

HIMEN # 07F0 : CLOAD « MAGNETO ».

donc, une fois le programme entré et sauvegardé, tapez CALL # 07FF pour enregistrer (Ready s'affiche quand c'est terminé) et CALL # 0900 pour écouter.

Le message enregistré en mémoire occupe l'espace entre # 1000 et # B400.

Vous pouvez également le transférer sur k7 en faisant :

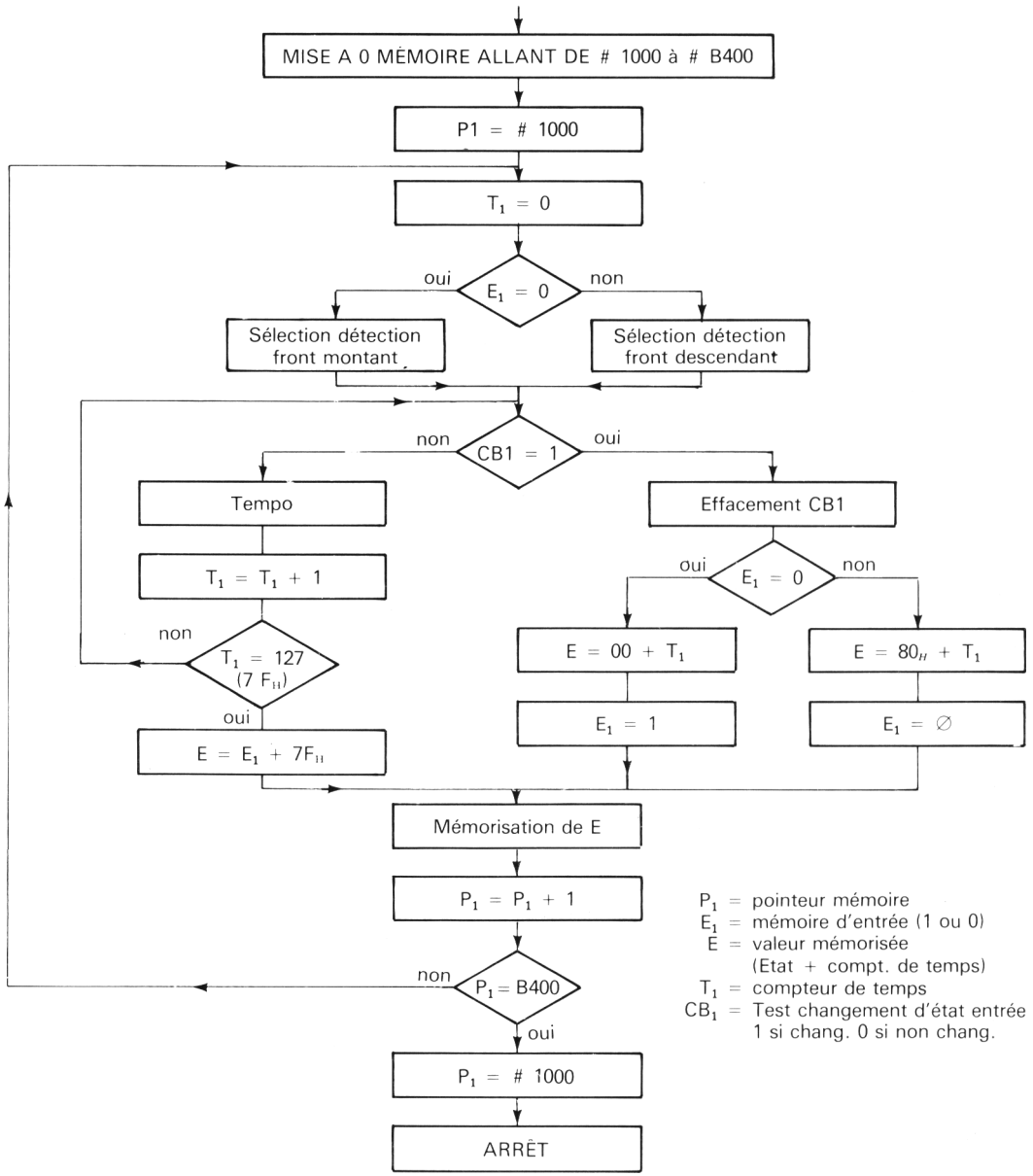
CSAVE « MESSAGE », A # 1000, E # B400.

pour garder l'ensemble programme + message, il faudra faire :

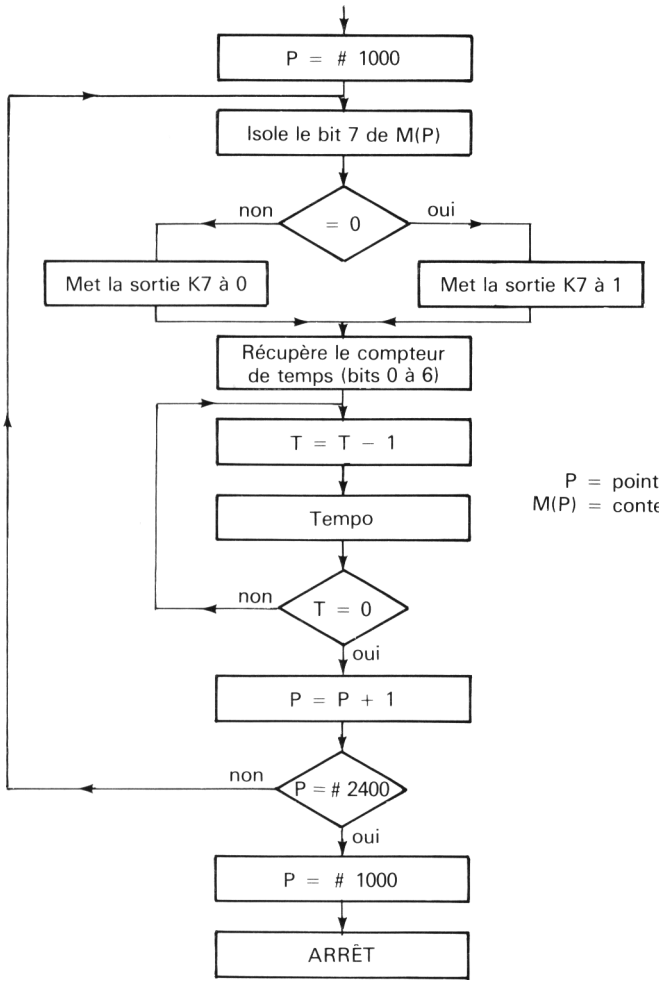
CSAVE « PARLE », A # 07FF, E # B400.

Au chargement : HIMEM # 07F0 : CLOAD « PARLE ».

SYNOPTIQUE PROGRAMME ENREGISTREMENT



SYNOPTIQUE PROGRAMME LECTURE

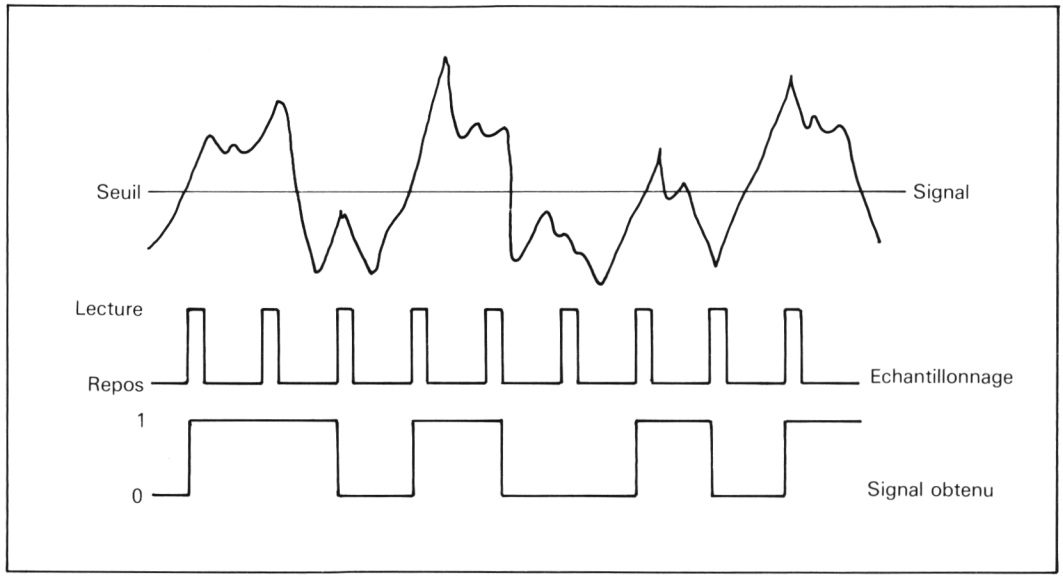


P = pointeur mémoire
M(P) = contenu mémoire pointée par P

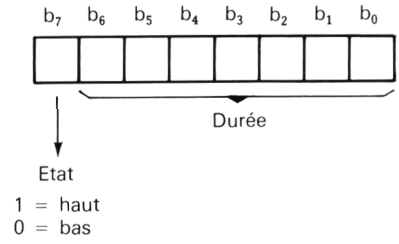
FONCTIONNEMENT

Il s'agit d'un échantillonnage du signal entrant. Si le niveau dépasse un certain seuil, on a 1 sinon c'est un 0.

Plus l'échantillonnage est rapide, meilleure est la qualité.
Le programme mesure la durée des états hauts et bas du signal résultant.



FORMAT DE LA MÉMOIRE :



MAGNÉTOPHONE NUMÉRIQUE AVEC ORIC-1

ENREGISTREMENT

```
FD
07FF 78       SEI
0800 A900     LDA %#00
0802 8D0010   STA #1000
0805 EE0308   INC #0803
0808 D0F8     BNE #0802
080A EE0408   INC #0804
080D AD0408   LDA #0804
0810 C9B4     CMP %#B4
0812 D0EC     BNE #0800
0814 A900     LDA %#00
0816 8DF107   STA #07F1
0819 A900     LDA %#00
081B 8DF207   STA #07F2
081E ADF107   LDA #07F1
0821 F009     BEQ #082C
0823 A9CD     LDA %#CD
0825 8D0C03   STA #030C
0828 A900     LDA %#00
082A F005     BEQ #0831
082C A9DD     LDA %#DD
082E 8D0C03   STA #030C
0831 AD0D03   LDA #030D
0834 2910     AND %#10
0836 F024     BEQ #085C
0838 A910     LDA %#10
083A 8D0D03   STA #030D
083D ADF107   LDA #07F1
0840 D00D     BNE #084F
0842 A900     LDA %#00
0844 0DF207   ORA #07F2
0847 A280     LDX %#80
0849 8EF107   STX #07F1
084C 4C7108   JMP #0871
084F A980     LDA %#80
0851 0DF207   ORA #07F2
0854 A200     LDX %#00
```

Inhibe les interruptions

Remplissage mémoire de 0

E1 = 0

T1 = 0

Si E1 = 0 saut en 082C

Sélection CB1 actif sur front descendant

Saut en 0831

Sélection CB1 actif sur front montant

Lecture entrée
Si 0 saut en 085C
Effacement CB1

Si E1 = 1 saut en 084F

A = T1

X = 80

E1 = 80
Saut en 0871

A = 80 + T1

```
0856 8EF107 STX #07F1
0859 4C7108 JMP #0871
085C A001   LDY %#01
085E 88     DEY
085F D0FD   BNE #085E
0861 EEF207 INC #07F2
0864 ADF207 LDA #07F2
0867 C97F   CMP %#7F
0869 D0C6   BNE #0831
086B ADF107 LDA #07F1
086E 0DF207 ORA #07F2
0871 8D0010 STA #1000
0874 EE7208 INC #0872
0877 D0A0   BNE #0819
0879 EE7308 INC #0873
087C AD7308 LDA #0873
087F C9B4   CMP %#B4
0881 D096   BNE #0819
0883 A910   LDA %#10
0885 8D0408 STA #0804
0888 8D7308 STA #0873
088B 58     CLI
088C 60     RTS
```

E1 = 0
Saut en 0871

Tempo

T1 = T1 + 1

Bouclè à lecture entrée si T1 < 7F

Mémorise E

Arrêt si fin mémoire

```
09
0900 78     SEI
0901 AD0010 LDA #1000
0904 48     PHA
0905 2980   AND %#80
0907 F008   BEQ #0911
0909 A2B0   LDX %#B0
090B 8E0003 STX #0300
090E 4C1609 JMP #0916
0911 A230   LDX %#30
0913 8E0003 STX #0300
0916 68     PLA
0917 297F   AND %#7F
0919 A8     TAY
```

LECTURE

Prend la donnée

isole le bit 7
saut si 0 en 0911
Met PB7 à 1

Saut en 0916
Met PB7 à 0

Isole le compteur de temps

```
091A C8      INY
091B A204    LDX %#04
091D CA      DEX
091E D0FD    BNE #091D
0920 88      DEY
0921 D0F8    BNE #091B
0923 EE0209  INC #0902
0926 D0D9    BNE #0901
0928 EE0309  INC #0903
092B AD0309  LDA #0903
092E C9B4    CMP %#B4
0930 D0CF    BNE #0901
0932 A910    LDA %#10
0934 8D0309  STA #0903
0937 58      CLI
0938 60      RTS
```

Tempo

Boucle si compteur # 0

Arrêt si fin mémoire

ANNEXE

PROGRAMMES MODIFIÉS POUR ATMOS

Les programmes dont les listings ne sont pas reproduits dans cette section du livre fonctionnent sans modification.

PROGRAMME RTTY ATMOS

La version proposée est une version simplifiée (donc plus courte à introduire) de la version ORIC-1. Nous avons supprimé la fonction « boîtes à lettres » mais le lecteur désireux de l'ajouter pourra s'aider du listing ORIC-1 (BASIC et ASSEMBLEUR) et de ses commentaires pour intégrer cette possibilité qui, soulignons le, n'appelle pas de modifications d'adresses.

Les autres caractéristiques du programme sont identiques.

```
 1  REM      ***** RTTY *****
 2  REM      *  @ D.BUNOMO    *
 3  REM      *  & E.DUTERTRE  *
 4  REM      *  F1EZH   F6GKQ *
 5  REM      *    29-02-1984  *
 6  REM      *        V.01    *
 7  REM      *     A T M O S  *
 8  REM      ******************
```

```
9 REM
10 HIMEM#7000
12 CLS:PRINT:PRINT:PRINT"PATIENTEZ QUELQUES INSTANTS"
15 DIMME$(4)
20 FORA=#9120TO#915A:READD:POKEA,D:NEXT
25 FORA=#9000TO#9061:READD:POKEA,D:NEXT
30 FORA=#9410TO#95BA:READD:POKEA,D:NEXT
35 FORA=#9160TO#920F:READD:POKEA,D:NEXT
40 FORA=#9610TO#964B:READD:POKEA,D:NEXT
49 REM--- TRANSCODAGE EMISSION ---
50 DATA#04,#00,#05,#00,#00,#1A,#00,#05,#0F,#12,#09,#11,#0C,#03,#1C,#1D,#16
52 DATA#17,#13,#01,#0A,#10,#15,#07,#06
54 DATA#18,#0E,#00,#00,#1E,#00,#19,#00,#03,#19,#0E,#09,#01,#0D,#1A,#14,#06
56 DATA#0B,#0F,#12,#1C,#0C,#18,#16,#17,#0A,#05,#10,#07,#1E,#13,#1D,#15,#11
58 DATA#00,#05,#2B,#31,#14,#C5,#00,#00,#00,#00,#00,#00,#00,#00,#00,#00
59 REM--- GENERATION TONALITES ---
60 DATA#98,#48,#78,#HE,#02,#90,#A9,#00,#20,#90,#F5
```

```
61 DATA#20,#55,#90,#A9,#05,#8D,#01,#90
62 DATA#EA,#6E,#00,#90,#B0,#1F,#AE,#02,#90,#A9,#00,#20,#90,#F5,#20,#55,#90
64 DATA#CE,#01,#90,#D0,#EB,#AE,#03,#90,#A9,#00,#20,#90,#F5,#20,#51,#90,#58
66 DATA#68,#A8,#60,#AE,#03,#90,#4C,#2C,#90
68 DATA#EA,#EA,#EA,#A0,#1E,#D0,#03,#A0,#14,#EA,#A2,#C7,#CA,#D0,#FD
70 DATA#88,#D0,#F8,#EA,#60
100 REM---- PARTIE RECEPTION ----
110 DATA#A9,#00,#85,#3B,#A9,#E7,#8D,#02,#03,#EA,#EA,#EA,#EA,#EA
120 DATA#58,#AD,#08,#02,#C9,#38,#F0,#05,#85,#35,#4C,#B0,#94
130 DATA#78,#AD,#00,#03,#29,#10,#D0,#EB,#A9,#05,#85,#39
140 DATA#20,#90,#94,#A9,#00,#85,#3A,#20,#94,#94,#A9,#10,#2D,#00,#03
150 DATA#05,#3A,#85,#3A,#C6,#39,#F0,#08,#46,#3A
160 DATA#4C,#3E,#94,#EA,#EA,#EA
170 DATA#18,#A5,#3A,#65,#3B,#85,#3A,#A6,#3A,#BD,#00,#95
175 DATA#EA,#EA,#EA,#EA,#EA
180 DATA#C9,#FF,#D0,#08,#A9,#20,#85,#3B,#4C,#86,#94,#EA
190 DATA#C9,#FE,#D0,#07,#A9,#00,#85,#3B,#4C,#86,#94,#20,#50,#95
200 DATA#AA,#20,#7C,#F7,#EA
210 DATA#AD,#00,#03,#29,#10,#F0,#F9,#4C,#1E,#94
220 DATA#A0,#0A,#D0,#03,#A0,#14,#EA,#A2,#C7,#CA
230 DATA#D0,#FD,#88,#D0,#F8,#EA,#60
235 DATA#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA
237 REM---- OPTIONS CLAVIER ----
240 DATA#A5,#35,#C9,#BA,#D0,#07,#A9,#20,#85,#3B,#4C,#2B,#94
245 DATA#C9,#8F,#D0,#07,#A9,#00,#85,#3B,#4C,#2B,#94
250 DATA#C9,#B6,#D0,#08,#A2,#0C,#20,#7C,#F7,#4C,#2B,#94
255 DATA#C9,#82,#D0,#06,#20,#79,#95,#4C,#2B,#94
260 DATA#C9,#9E,#D0,#05,#A9,#01,#85,#45,#60
265 DATA#C9,#89,#D0,#05,#A9,#00,#85,#45,#60,#4C,#2B,#94
270 DATA#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA
349 REM--- TRANSCODAGE RECEPTION ---
350 DATA#20,#45,#0A,#41,#20,#53,#49,#55,#0D,#44,#52,#4A,#4E,#46,#43,#4B,#54
360 DATA#5A,#4C,#57,#48,#59,#50,#51,#4F,#42,#47,#FF,#4D,#58,#56,#FE
370 DATA#21,#33,#0A,#2D,#20,#22,#38,#37,#0D,#2A,#34,#0A,#2C,#45,#3A,#28,#35
380 DATA#2B,#29,#32,#48,#36,#30,#31,#39,#3F,#25,#FF,#2E,#2F,#3D,#FE
385 DATA#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA
390 REM---- MEMOR TEXTE ----
400 DATA#85,#3A,#18,#A9,#01,#65,#3D,#85,#3D,#A9,#00,#65,#3E,#85,#3E
405 DATA#A2,#01,#A1,#3C,#C9,#99,#D0,#08,#A9,#00,#85,#3D
410 DATA#A9,#80,#85,#3E,#A5,#3A,#81,#3C,#60
415 DATA#EA,#EA,#EA,#EA,#EA,#AD,#08,#02,#C9,#38,#D0,#F9
420 DATA#A2,#0C,#20,#7C,#F7,#18,#A9,#01,#65,#3F,#85,#3F,#A9,#00
425 DATA#65,#40,#85,#40,#A2,#01,#A1,#3E,#C9,#99,#D0,#09
430 DATA#A9,#00,#85,#3F,#A9,#80,#85,#40,#60
435 DATA#AA,#20,#7C,#F7,#AD,#08,#02,#C9,#38,#F0,#D7
440 DATA#C9,#84,#F0,#F5,#A2,#0C,#20,#7C,#F7,#4C,#9A,#95,#EA
499 REM---- SAISIE MACHINE -----
500 DATA#A9,#00,#A8,#20,#E8,#C5,#8D,#06,#90,#C8,#99,#00,#93
```

```
505 DATA#EA,#EA,#EA,#EA,#EA,#EA
510 DATA#C9,#40,#F0,#04,#C9,#5E,#D0,#07,#8D,#00,#91,#EA,#EA,#EA,#60
515 DATA#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA,#EA
520 DATA#AA,#20,#7C,#F7,#C9,#0D,#D0,#06,#20,#F0,#91,#4C,#60,#91
525 DATA#AA,#20,#AD,#91,#CE,#02,#91,#D0,#03,#20,#F0,#91,#4C,#60,#91
527 DATA#C9,#20,#F0,#31
530 DATA#C9,#40,#B0,#1B,#AD,#03,#91,#D0,#28,#A9,#01,#8D,#03,#91
535 DATA#A9,#1B,#8D,#00,#90,#20,#10,#90,#4C,#E2,#91,#EA,#EA,#EA,#EA,#EA
540 DATA#EA,#AD,#03,#91,#F0,#0E,#A9,#00,#8D,#03,#91,#A9,#1F,#8D,#00,#90
545 DATA#20,#10,#90,#AE,#06,#90,#BD,#00,#91,#8D,#00,#90,#20,#10,#90,#60,#EA
550 DATA#A2,#0A,#20,#7C,#F7,#AD,#01,#91,#8D,#02,#91,#A9,#08,#8D,#00,#90
555 DATA#20,#10,#90,#A9,#02,#8D,#00,#90,#20,#10,#90,#60,#EA,#EA,#EA,#EA
599 REM----- LEC. MES. MEMO.-----
600 DATA#AC,#07,#90,#B9,#00,#75,#8D,#06,#90,#C9,#40,#F0,#04
605 DATA#C9,#5E,#D0,#07,#8D,#00,#91,#EA,#EA,#EA,#60
610 DATA#AA,#20,#7C,#F7,#C9,#0D,#D0,#06,#20,#F0,#91
615 DATA#4C,#41,#96,#20,#AD,#91,#CE,#02,#91,#D0,#03
620 DATA#20,#F0,#91,#EE,#07,#90,#D0,#03,#EE,#15,#96,#4C,#10,#96
999 REM-------------------------------
1000 REM FORA=#8000TO#8FFF:POKEA,32:NEXT
1002 INK7:PAPER0
1003 POKE#45,1
1005 POKE#3D,00:POKE#3E,128:POKE#3F,00:POKE#40,128:PUKE#8FFF,153
1008 GOSUB6000
1010 PRINTCHR$(6):PRINTCHR$(12)
1015 POKE#9100,64:POKE#9103,0
1020 L=1:C=0
1050 REM----- PRESENTATION -----
1060 PRINTTAB(9);CHR$(4);CHR$(27);"J-----RTTY-----";CHR$(4)
1065 PRINT
1070 PRINTTAB(9);CHR$(27);"LF1EZH--F6GKQ"
1080 FORA=#BBA4TO#BBA7:POKEA,32:NEXT
1100 REM----- REINITIALISATION -----
1105 PRINT:PRINT:INPUT"NB. DE COLONNES POUR CR AUTO :";K
1106 POKE#9101,K:POKE#9102,K
1107 IFK=0THENNRC=1ELSENRC=0
1110 PRINT:PRINT:INPUT"VITESSE (45 OU 50 BAUDS):";V
1115 IFV<>50THEN1119
1116 REM----- 50 BDS -----
1117 POKE#9491,10:POKE#9495,20:POKE#9056,20
1118 POKE#9498,199:POKE#9059,199:POKE#9052,30:GOTO1125
1119 REM----- 45 BDS -----
1120 POKE#9491,11:POKE#9495,22:POKE#9056,22
1121 POKE#9498,199:POKE#9059,199:POKE#9052,33
1125 PRINT:PRINT:INPUT"SHIFT (+/-HERTZ):";S
1130 PRINT:PRINT:PRINT:PRINT" E - Pour Emettre ":PRINT" R - Pour Recevoir"
1135 POKE#26A,43
1140 GETA$:IFA$="E"THEN2000
```

```
1150 IFA$<>"R"THEN1140
1299 REM---- S/P DE RECEPTION ----
1300 POKE#27E,27:DOKE#27C,1040: CLS:SOUND1,49,0
1302 POKE#302,231
1305 FORA=#BB8BTO#BB9A:POKEA,32:NEXT
1320 PRINT@2,20;"    C - Force mode CHIFFRES"
1325 PRINT@2,21;"    L - Force mode LETTRES"
1330 PRINT@2,22;"    S - Efface l'ecran"
1335 PRINT@2,23;"    M - Affiche la memoire"
1340 PRINT@2,24;"    E - Emission ";S;" HZ  ";V;" BDS"
1345 PRINT@2,25;"    T - Emission avec modif. Parametres"
1405 POKE#27E,18:DOKE#27C,720
1415 CLS
1420 CALL#9410
1500 POKE#27E,27:DOKE#27C,1040
1505 A$=KEY$
1510 CLS
1600 IFPEEK(#45)=0THEN1010ELSE2000
1999 REM---- S/P D'EMISSION ----
2000 CLS
2002 POKE#27E,12:DOKE#27C,480
2007 PRINT@2,14;"@ POUR PASSER EN RECEPTION"
2008 PRINT@2,16;"^ POUR CHANGER OPTION EMISSION"
2009 PRINT
2010 PRINT@2,18;"0 Emission a Partir du clavier"
2020 PRINT@2,19;"1 Emission du messa9e d'aPPel"
2025 PRINT@2,20;"2 Emission du messa9e de test"
2030 PRINT@2,21;"3 Emission du messa9e memorise"
2035 PRINT@2,22;"4 Emission d'une li9ne de RY"
2050 PRINT@17,24;"-> VOTRE CHOIX ? ";
2052 GETRE$
2053 PRINT@17,24;"                  "
2056 IF(RE$<>"@")OR(RE$<>"^")THENRE=VAL(RE$)
2057 IFRE$="@"THEN1300
2058 IFRE$="^"THEN2330
2060 CLS:GOSUB2100:GOTO2230
2099 REM---- PREPARE EMISSION ----
2100 A$=" F1EZH - F6GKQ "
2110 FORA=1TOLEN(A$):POKE#BB8A+A,ASC(MID$(A$,A,1)):NEXT
2150 IFS>0THENQ=1275:W=Q+S
2160 IFS<0THENW=1275:Q=W-S
2180 H=INT(62500/Q):B=INT(62500/W)
2190 POKE#9003,H:POKE#9002,B
2200 SOUND1,H,5
2205 POKE#302,167
2210 RETURN
2230 IFRE<>0THEN2250
2235 CALL#9160
```

```
2240 GOTO2300
2250 RE=RE-1
2255 DOKE#9614,(29952+(256*RE))
2257 POKE#9007,0
2260 CALL#9610
2300 IFPEEK(#9100)=#40THEN1300
2305 IFPEEK(#9100)=#5ETHEN2330
2330 GOTO2050
2620 GOTO2230
3220 IF(FCE<SUP)AND(FCE>INF)THENFCE=1ELSEFCE=0
5999 REM---- MESSAGES MEMORISES ----
6000 RY$="RYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRY"
6010 AP$="    CQ CQ CQ DE F6GKQ F6GKQ F6GKQ / 91    "
6020 TE$="  TEST TEST DE F6GKQ / 91 TEST TEST      "
6030 CH$="    0123456789 ' * - , . ( ) + / = % :    "
6040 LE$="         QWERTYUIOPASDFGHJKLZXCVBNM        "
6050 CQ$=" MICRO-COMPUTER ORIC-1.SOFT F1EZH-F6GKQ "
6060 ST$="TX MULTI 750E,DECODEUR PLL,ANT 14EL/22M "
6100 ME$(1)=CHR$(13)+RY$+AP$+AP$+AP$+RY$+"^"
6110 ME$(2)=CHR$(13)+RY$+TE$+CH$+LE$+RY$+"^"
6120 ME$(3)=CHR$(13)+CQ$+ST$+"^"
6130 ME$(4)=CHR$(13)+RY$+"^"
6200 FORI=0TO3
6210 A=#7500+(I*256)
6215 POKEA,13
6220 GOSUB6300:NEXT
6230 RETURN
6300 J=1
6310 REPEAT
6320 C$=MID$(ME$(I+1),J,1)
6330 POKE(A+J),ASC(C$)
6340 J=J+1
6350 UNTILC$="^"
6360 RETURN
```

PROGRAMME CONTEST ATMOS

La nouvelle version du programme CONTEST tient compte du fait que l'ATMOS permet la sauvegarde et la lecture de fichiers. Néanmoins, à cause d'une erreur probable dans son BASIC, il affichera souvent ERRORS FOUND après leur chargement. Ne pas en tenir compte.

```
1  REM  +++++  CONTEST  +++++
2  REM  +                    +
3  REM  +(c) D. BUONOMO       +
4  REM  +      F6GKQ         +
5  REM  +    04-03-1984      +
6  REM  +       V.01         +
7  REM  +    A T M O S       +
8  REM  +                    +
9  REM  ++++++++++++++++++++++
```

```
10 REM
50 GOTO1000
69 REM---- CALCUL DE DISTANCE ----
70 IFASC(Q$)<84THEN100
80 A=-91+ASC(Q$):GOTO110
100 A=-65+ASC(Q$)
110 Q$=RIGHT$(Q$,4):B=-65+ASC(Q$)
130 Q$=RIGHT$(Q$,3):C=-48+ASC(Q$)
150 Q$=RIGHT$(Q$,2):D=-48+ASC(Q$)
170 Q$=RIGHT$(Q$,1):E=ASC(Q$)
190 IFD<>0THEN220
200 D=10:C=C-1
220 IFE=65THENE=3.1
230 IFE=66THENE=1.1
240 IFE=67THENE=1.3
250 IFE=68THENE=1.5
260 IFE=69THENE=3.5
270 IFE=70THENE=5.5
280 IFE=71THENE=5.3
290 IFE=72THENE=5.1
300 IFE=74THENE=3.3
310 H=INT(E):K=ABS(H-E)*10
330 GB=(2*A)+(D/5)-(H/30)
340 LB=41+B-(C/8)-(K/48)
390 IFINI=0THENRETURN
570 DG=GA-GB
580 A=SIN(LA/180*PI):B=SIN(LB/180*PI)
```

```
600 C=COS(LA/180*PI):D=COS(LB/180*PI)
620 E=COS(DG/180*PI):X=(A*B)+(C*D*E)
630 ACSX=-ATN(X/SQR(-X*X+1))+1.5708
635 DIST=111.323*(ACSX/PI*180)
645 IF(DIST-INT(DIST))>=.5THENDIST=1+INT(DIST)ELSEDIST=INT(DIST)
650 RETURN
999 REM---- INITIALISATIONS ----
1000 HIMEM#97FF
1015 INK7:PAPER0
1020 DOKE18,48031:PRINT"Contest"
1050 CLS
1055 PRINT:PRINT:PRINT:PRINT
1060 INPUT"VOTRE LOCATOR ";Q$
1062 QQ$=Q$
1063 PRINT:PRINT
1065 INPUT"VOTRE INDICATIF ";I$
1070 GOSUB70
1075 LA=LB:GA=GB
1080 INI=1
1150 DIMU$(505),V$(501)
1200 POKE#26A,35
1210 PRINT:PRINT:PRINT
1220 PRINT" AVEZ-VOUS DEJA SAUVEGARDE DES DONNEES ?"
1230 GETRR$
1240 IFRR$<>"O"THENDER=1:GOTO2000
1245 REM---- RELECTURE DU FICHIER ----
1250 CLS:PRINT:PRINT:PRINT:PRINT
1260 PRINT" REPOSITIONNEZ LA CASSETTE DE DONNEES"
1270 PRINT:PRINT:PRINT" PRESSEZ LA TOUCHE PLAY DU MAGNETO "
1280 PRINT:PRINT:PRINT" ----- PRESSEZ UNE TOUCHE -----"
1290 GETRR$:PING
1300 PRINT:PRINT:PRINT"  PATIENTEZ QUELQUES INSTANTS"
1400 RECALLU$,"INDICS"
1410 RECALLV$,"DONNEES"
1420 DER=VAL(MID$(U$(501),2,LEN(U$(501))))
1430 CUM=VAL(MID$(U$(502),2,LEN(U$(502))))
1440 MOY=VAL(MID$(U$(503),2,LEN(U$(503))))
2000 PING:CLS
2100 PRINT:PRINT"TOTAL CUMULE  :  ";CUM
2102 PRINT"MOYENNE QSO   :  ";MOY
2104 PRINT"LE PROCHAIN QSO SERA No  :  ";DER
2105 PRINT:PRINT
2110 IFDER>500THENPRINT"MAXIMUM DE QSO ATTEINT ":GOTO2125
2120 PRINT:PRINT:PRINT"INTRODUISEZ L'INDICATIF OU:"
2125 PRINT
2130 PRINT"   - S POUR SAUVEGARDER FICHIERS"
2135 PRINT"   - E POUR LISTER SUR ECRAN"
2140 PRINT"   - P POUR LISTER SUR IMPRIMANTE"
```

```
2145 PRINT:INPUT"       -> VOTRE CHOIX ";R$
2150 IFR$="S"THEN5000
2155 IFR$="E"THEN3000
2160 IFR$="P"THEN4000
2165 IFDER>500THENCLS:PING:GOTO2110
2170 I=0
2200 REPEAT
2210 I=I+1
2220 IFR$<>MID$(U$(I),1,LEN(R$))THEN2300
2230 CLS
2235 PRINT:PRINT:PRINTR$;" DEJA CONTACTE,QSO NO ";I
2240 PRINT:PRINT:PRINTU$(I);SPC(12-LEN(U$(I)));
2245 PRINTV$(I)
2250 PRINT:PRINT:PRINT:GOTO2100
2300 UNTILI=DER
2400 CLS
2410 PRINT:PRINT:PRINT:PRINT
2415 PRINT"POUR QSO NO ";DER;" VOULEZ-VOUS ";R$;" ?";
2420 GETRR$:PRINTRR$
2425 IFRR$<>"O"THENGOTO2100
2450 PRINT:PRINT:PRINT
2455 INPUT"INDICATIF COMPLET ";U$(DER)
2332 PRINT
2465 INPUT"QTR DU CONTACT (HHMM) ";QTR$
2470 PRINT
2475 INPUT"QRA LOCATOR ";QRA$
2480 PRINT
2485 INPUT"GROUPE PASSE ";GP$
2490 PRINT
2495 INPUT"GROUPE RECU ";GR$
2496 PING
2497 PRINT:PRINT" A POUR ANNULER V POUR VALIDER ";:GETRR$:IFRR$="A"THEN2400
2500 PRINT
2505 Q$=QRA$:GOSUB70
2510 CLS:PRINT:PRINT
2515 PRINTU$(DER);:PRINTSPC(12-LEN(U$(DER)));
2530 V$(DER)=QTR$+" "+QRA$+" "+GP$+" "+GR$+" "+STR$(DIST)
2535 PRINTV$(DER):PRINT
2540 CUM=CUM+DIST
2545 MOY=CUM/DER
2570 DER=DER+1
2600 GOTO2100
2999 REM---- LISTAGE SUR ECRAN ----
3000 CLS
3005 PRINT:PRINT:INPUT"LISTAGE A PARTIR DU NO ";N
3010 FORI=NTODER
3020 IFPEEK(#208)=#84THENN=DER:GOTO2120
```

```
3100 PRINTU$(I)SPC(12-LEN(U$(I)))V$(I)
3150 NEXT
3200 GOTO2120
3999 REM---- LISTAGE SUR IMPRIMANTE----
4000 CLS:LPRINT
4005 PRINT:PRINT:INPUT"LISTAGE A PARTIR DU NO ";N
4012 LPRINT:LPRINTCHR$(14);"STATION : ";I$:LPRINT"LOCATOR : ";QQ$
4013 LPRINTCHR$(15):LPRINT
4015 FORI=NTODER
4020 IFPEEK(#208)=#84THENN=DER:GOTO4160
4100 LPRINTU$(I)SPC(12-LEN(U$(I)))V$(I)
4105 PRINTU$(I)SPC(12-LEN(U$(I)))V$(I)
4150 NEXT
4155 LPRINTCHR$(14):LPRINT
4160 LPRINT"No DERNIER QSO : ";DER-1
4162 LPRINT"TOTAL CUMULE   : ";CUM
4164 LPRINT"MOYENNE PAR QSO: ";MOY
4165 LPRINTCHR$(15)
4200 GOTO2120
4999 REM---- SAUVEGARDE FICHIERS ----
5000 REM
5020 U$(501)=STR$(DER):U$(502)=STR$(CUM):U$(503)=STR$(MOY)
5250 CLS:PRINT:PRINT:PRINT:PRINT
5260 PRINT" REPOSITIONNEZ LA CASSETTE DE DONNEES"
5270 PRINT:PRINT:PRINT" PRESSEZ LA TOUCHE RECORD DU MAGNETO"
5280 PRINT:PRINT:PRINT" ----- PRESSEZ UNE TOUCHE -----"
5290 GETRR$:PING
5300 PRINT:PRINT:PRINT"  PATIENTEZ QUELQUES INSTANTS"
5400 STOREU$,"INDICS"
5410 STOREV$,"DONNEES"
5500 PING:CLS:GOTO2120
19999 END
```

```
 1  REM  ***** ATTENUATEURS *****
 2  REM  *                      *
 3  REM  *    @ Denis BONOMO    *
 4  REM  *          F6GKQ       *
 5  REM  *     03-03-1984 V.01  *
 6  REM  *       A T M O S      *
 7  REM  *                      *
 8  REM  ************************
```

```
9 REM
1000 REM---- ATTENUATEURS PI/TE ----
1010 GOSUB1500
1020 CLS:PRINT:PRINT:PRINT
1025 INPUT"ATTENUATON DESIREE (DB) ";A
1030 N=EXP((A*2.302585/20))
1035 PRINT:PRINT
1040 INPUT"IMPEDANCE CARACTERISTIQUE (OHMS) ";ZC
1045 PRINT:PRINT
1100 Z1=ZC*((N*N-1)/(2*N))
1110 Z2=ZC*((N+1)/(N-1))
1120 Z1=INT(Z1*10)/10
1130 Z2=INT(Z2*10)/10
1150 Z3=ZC*((N-1)/(N+1))
1160 Z4=ZC*((2*N)/((N*N)-1))
1170 Z3=INT(Z3*10)/10
1180 Z4=INT(Z4*10)/10
1190 GOSUB1300
1200 PRINTCHR$(17)
1205 PRINT@8,15;Z1
1210 PRINT@26,15;Z3
1240 PRINT@7,20;Z2
1250 PRINT@30,20;Z4
1265 PRINT@2,25;"ATTENUATION DE ";A;" DB SOUS ";ZC;" OHMS"
1270 PRINTCHR$(17):PRINT"DESIREZ-VOUS UN AUTRE CALCUL (O/N) ?":GETR$
1275 IFR$="O"THEN1020ELSEEND
1300 REM---- DESSINE ----
1320 PRINT@2,17;"    ";"a";"i";"i";"f";"j";"g";"i";"i";"a"
1330 PRINT@2,18;"    ";"b";"       ";"b"
1340 PRINT@2,19;"    ";"d";"       ";"d"
1350 PRINT@2,20;"    ";"k";"       ";"k"
1360 PRINT@2,21;"    ";"e";"       ";"e"
1370 PRINT@2,22;"    ";"b";"       ";"b"
1380 PRINT@2,23;"    ";"c";"       ";"c"
1400 PRINT@23,17;"h";"f";"j";"g";"h";"a";"h";"f";"j";"g";"h"
1405 PRINT@28,18;"b"
```

```
1410 PRINT@28,19;"d"
1420 PRINT@28,20;"k"
1430 PRINT@28,21;"e"
1440 PRINT@28,22;"b"
1450 PRINT@28,23;"c"
1470 RETURN
1500 FORA=46856TO46943
1510 READD$:D=VAL("#"+D$)
1520 POKEA,D:NEXT
1530 RETURN
1600 DATA0C,0C,1E,3F,3F,1E,0C,0C
1605 DATA0C,0C,0C,0C,0C,0C,0C,0C
1610 DATA0C,0C,0C,0C,0C,0C,3F,3F
1615 DATA3F,21,21,21,21,21,21,21
1620 DATA21,21,21,21,21,21,21,3F
1625 DATA00,3F,20,20,20,20,3F,00
1630 DATA00,3F,01,01,01,01,3F,00
1635 DATA00,00,00,3F,3F,00,00,00
1640 DATA00,00,00,3F,3F,00,00,00
1645 DATA00,3F,00,00,00,00,3F,00
1650 DATA21,21,21,21,21,21,21,21
```

```
 1  REM   ++++++ MIRE ++++++
 2  REM   +                +
 3  REM   + (c) D. BONOMO  +
 4  REM   +      F6GKQ     +
 5  REM   +   03-03-1984   +
 6  REM   +     (V.01)     +
 7  REM   +    A T M O S   +
 8  REM   +                +
 9  REM   ++++++++++++++++++
```

```
10 CLS
40 FORI=48036TO48039:POKEI,32:NEXTI
45 PRINT
50 PRINT"MISE A L'HEURE DE L'HORLOGE"
55 PRINT"--------------------------"
60 PRINT
65 PRINT"INTRODUIRE LES HEURES ET LES MINUTES"
70 PRINT"AVEC UN PEU D'AVANCE SUR LE TOP "
75 PRINT
80 PRINT"APPUYEZ SUR UNE TOUCHE AU TOP"
85 PRINT
90 INPUT"     LES HEURES :";HH
95 INPUT"     LES MINUTES:";MM
97 CLS
100 DIMCL(8)
200 FORI=1TO8
205 READCL(I)
210 NEXTI
215 DATA144,148,145,149,146,150,147,151
220 A$=CHR$(27):C$=CHR$(4):S$=A$+"J"
250 FORI=46856TO46983
255 READDI
260 POKEI,DI
265 NEXTI
270 DATA1,1,1,1,1,1,1,1
272 DATA32,32,32,32,32,32,32,32
274 DATA63,63,0,0,0,0,0,0
276 DATA0,0,0,0,0,0,63,63
278 DATA63,63,1,1,1,1,1,1
280 DATA63,63,32,32,32,32,32,32
282 DATA63,63,12,12,12,12,63,63
284 DATA63,63,32,32,32,32,63,63
286 DATA63,63,1,1,1,1,63,63
288 DATA63,63,63,0,0,63,63,63
290 DATA7,7,7,7,7,7,7,7
```

```
292 DATA27,27,27,27,27,27,27,27
294 DATA21,21,21,21,21,21,21,21
296 DATA63,63,63,63,63,63,63,63
298 DATA32,32,32,32,32,32,63,63
300 DATA1,1,1,1,1,1,63,63
1000 PRINT"   ";
1005 FORI=1TO33:PRINTCHR$(100);:NEXTI
1010 PRINT"   "
1100 PRINT"   ";C$;CHR$(98);"            ";
1101 PRINTA$;"P";A$;"G";A$;"J";
1102 PRINT"HH:MM";"    ";
1103 PRINTA$;"W";A$;"@";"        ";
1104 PRINTA$;"J";
1105 PRINTCHR$(97)
1110 PRINTC$
1190 FORJ=1TO2
1200 PRINT"   ";
1205 FORI=1TO3:PRINTCHR$(255);:NEXTI
1210 PRINT" ";CHR$(97);
1215 PRINT"                              ";
1220 FORI=1TO3:PRINTCHR$(255);:NEXTI
1230 PRINT
1240 NEXTJ
1250 FORJ=1TO2
1255 PRINT"   ";CHR$(255);
1260 FORI=1TO8:PRINTCHR$(254);CHR$(254);CHR$(255);CHR$(255);:NEXTI
1270 PRINT
1275 NEXTJ
1300 FORI=1TO5
1305 PRINT"   ";
1310 FORJ=1TO8
1320 PRINTA$;CHR$(CL(J));"    ";
1330 NEXTJ
1340 PRINTA$;CHR$(151);
1345 PRINTCHR$(98)
1350 NEXTI
1390 PRINT"   ";CHR$(104);
1400 FORI=1TO31:PRINTCHR$(103);:NEXTI
1410 PRINTCHR$(105);" "
1420 PRINT"   ";
1430 FORI=1TO33:PRINTCHR$(106);:NEXTI
1435 PRINT"   "
1450 PRINT"   ";CHR$(104);
1460 FORI=1TO31:PRINTCHR$(103);:NEXTI
1470 PRINTCHR$(105);" "
1500 FORI=1TO5
1505 PRINT"   ";
```

```
1510 PRINTCHR$(255);CHR$(255);CHR$(255);
1515 PRINT"k";"k";"k";
1520 PRINT"l";"l";"l";"l";"l";
1525 PRINT"m";"m";"m";"m";"m";"m";"m";"m";"m";"m";"m";
1530 PRINT"l";"l";"l";"l";"l";
1535 PRINT"k";"k";"k";
1540 PRINTCHR$(255);CHR$(255);CHR$(255)
1545 NEXTI
1600 FORI=1TO3
1610 PRINT"  ";CHR$( 98);
1620 FORJ=8TO1STEP-1
1630 PRINTA$;CHR$(CL(J));"   ";
1640 NEXTJ
1645 PRINTA$;CHR$(151)
1650 NEXTI
1700 PRINTC$;
1710 PRINT"  ";CHR$( 98);"  ";
1715 PRINTCHR$(110);CHR$(110);CHR$(107);
1720 PRINTA$;"P";A$;"G";A$;"N F 6 G K 0 - 9 1    ";A$;"W";A$;"@";A$;"J";
1725 PRINTCHR$(97)
1730 PRINTC$
1740 PRINT"  ";"o";
1745 FORI=1TO31:PRINTCHR$(100);:NEXTI
1750 PRINT"p"
1790 GETK$
1795 PRINTCHR$(17);
1797 PLOT17,1,"    ":PLOT17,2,"    "
1800 MM$=STR$(MM)
1805 HH$=STR$(HH)
1807 PRINT@14,1;S$:PRINT@14,2;S$
1810 IFHH>9THEN1818
1812 PRINT@18,1;HH$:PRINT@18,2;HH$
1814 PRINT@18,1;"0":PRINT@18,2;"0"
1816 GOTO1820
1818 PRINT@17,1;HH$:PRINT@17,2;HH$
1820 IFMM>9THEN1828
1822 PRINT@21,1;MM$:PRINT@21,2;MM$
1824 PRINT@21,1;"0":PRINT@21,2;"0"
1826 GOTO1845
1828 PRINT@20,1;MM$:PRINT@20,2;MM$
1845 PRINT@20,1;":":PRINT@20,2;":"
1850 FORI=1TO60:WAIT99 :NEXTI
1855 WAIT40
1860 MM=MM+1
1865 IFMM<60THEN1800
1870 MM=0
1875 HH=HH+1
```

```
1880 IFHH<24THEN1800
1890 HH=0
1895 GOTO1800
1955 PRINTCHR$(17);
1960 GETK$
1965 PRINTCHR$(17)
```

```
 1  REM      +++ LOCATORIC   +++
 2  REM      +  © Denis BONOMO  +
 3  REM      +       F6GKQ      +
 4  REM      +     03-03-1984   +
 5  REM      +       V.01       +
 6  REM      +     A T M O S    +
 7  REM      ++++++++++++++++++++++
```

```
8 REM
9 CLS:TEXT:PRINT:PRINT
10 PRINT" 1 -POUR CALCULER DES DISTANCES"
12 PRINT"    EN FONCTION DU QTH LOCATOR"
14 PRINT
15 PRINT" 2 -POUR DETERMINER UN LOCATOR"
17 PRINT"    EN FONCTION DES COORDONNEES"
19 PRINT
20 INPUT"VOTRE CHOIX ";R
22 CLS
25 ONRGOTO4000,6000
69 REM+++TRANSFO LOC COORDONNEES+++
70 IFASC(Q$)<84THENGOTO100
80 A=-91+ASC(Q$)
90 GOTO110
100 A=-65+ASC(Q$)
110 Q$=RIGHT$(Q$,4)
120 B=-65+ASC(Q$)
130 Q$=RIGHT$(Q$,3)
140 C=-48+ASC(Q$)
150 Q$=RIGHT$(Q$,2)
160 D=-48+ASC(Q$)
170 Q$=RIGHT$(Q$,1)
180 E=ASC(Q$)
190 IFD<>0THEN220
200 D=10
210 C=C-1
220 IFE=65THENE=3.1
230 IFE=66THENE=1.1
240 IFE=67THENE=1.3
250 IFE=68THENE=1.5
260 IFE=69THENE=3.5
270 IFE=70THENE=5.5
280 IFE=71THENE=5.3
290 IFE=72THENE=5.1
300 IFE=74THENE=3.3
310 H=INT(E)
```

```
320 K=ABS(H-E)*10
330 GB=(2*A)+(D/5)-(H/30)
340 LB=41+B-(C/8)-(K/48)
370 PRINT"LAT:";LB;TAB(20);"LON:";GB
390 RETURN
400 CLS
420 INPUT"VOTRE LOCATOR ";Q$
440 GOSUB70
450 X0=121:Y0=54
460 CURSETX0,Y0,1
470 LA=LB:GA=GB
500 INPUT"LOCATOR DU CORRESPONDANT ";Q$
520 GOSUB70
569 REM+++CALCUL DISTANCE+++
570 DG=GA-GB
580 A=SIN(LA/180*PI)
590 B=SIN(LB/180*PI)
600 C=COS(LA/180*PI)
610 D=COS(LB/180*PI)
620 E=COS(DG/180*PI)
625 X=(A*B)+(C*D*E)
630 ACSX=-ATN(X/SQR(-X*X+1))+1.5708
635 DIST=111.323*(ACSX/PI*180)
640 PRINT"DIST: ";INT(DIST*10)/10,
650 DC=DIST
669 REM+++CALCUL AZIMUT+++
670 DIST=DIST/1.852
680 R=DIST/60
690 F=COS(R/180*PI)
700 G=SIN(R/180*PI)
710 X=(B-F*A)/(G*C)
715 ACSX=-ATN(X/SQR(-X*X+1))+1.5708
720 AZI=ACSX/PI*180
730 IFGA-GB>0THENAZI=360-AZI
740 PRINT"AZI: ";INT(AZI*10)/10;
749 REM+++TRACE DU PARCOURS+++
750 XA=(DC*SIN(AZI/180*PI))/5.6
760 YA=(DC*COS(AZI/180*PI))/5.6
765 IF(X0+XA)>=0AND(X0+XA)<=239AND(Y0-YA)>=0AND(Y0-YA)<=199THEN775
770 XA=0:YA=0:GOTO790
775 PATTERN170
780 DRAWXA,-YA,1
790 CURSETX0,Y0,1
850 IFKEY$=""THEN850ELSECLS
860 DRAWXA,-YA,0:CURSETX0,Y0,1
870 GOTO500
3999 REM+++DESSIN DE LA CARTE+++
```

```
4000 HIRES
4010 CURSET115,5,1
4020 FORI=1TO54
4030 READX,Y
4040 DRAWX,Y,1
4050 NEXTI
4900 GOTO400
5000 DATA5,8,21,15,5,-6,4,8,17,2,12,4,17,4,-7,31,-16,18,-3,13,9,-4,6,8,3,29
5010 DATA8,5,2,8,-19,15,-11,-3,-4,-4,-5,4,-9,-3,-14,10,2,11
5020 DATA-17,5,-15,-8,-17,2,-22,-12,8,-43,8,9,-8,-16,-2,-9,-8,-4,-5,-14,5,2
5030 DATA-10,-8,-11,-7,-17,-2,-3,-5,5,-2,-4,-2,4,-2,-5,-2,5,-5, 17,-3
5040 DATA8,4,15,-3,-4,-13,-2,-10,9,0,0,6,22,-2,-3,-5,15,-8,0,-16,10,-2
5999 REM+++DETERMINE QTH LOCATOR+++
6000 DIMT$(3,3)
6002 TT$="FEDGJCHAB"
6003 K=1
6005 FORI=1TO3
6010 FORJ=1TO3
6015 T$(I,J)=MID$(TT$,K,1)
6016 K=K+1
6020 NEXTJ
6025 NEXTI
6050 PRINT:PRINT
6055 PRINT"COORDONNEES EN DEGRES DECIMAUX"
6100 INPUT"LAT: ";LA
6105 INPUT"LON: ";GA
6110 PRINT
6119 REM+++LATITUDE+++
6120 B$=CHR$(INT(LA)+25)
6125 M=((LA-INT(LA))*60)/7.5
6150 C=7-INT(M)
6155 C$=STR$(C)
6170 EL=3*(M-INT(M))
6175 IFEL>1THEN6185
6180 EL=1:GOTO6300
6185 IFEL>2THEN6195
6190 EL=2:GOTO6300
6195 EL=3
6299 REM+++LONGITUDE+++
6300 GG=ABS(GA)
6305 G=(INT(GG)*60)+(GG-INT(GG))*60
6310 G=G/120
6320 IFGA<0THEN6335
6325 A$=CHR$(65+INT(G))
6330 GOTO6340
6335 A$=CHR$(90-INT(G))
6340 G=(G-INT(G))*10
```

```
6350 IFGA<0THEN6365
6355 D=1+INT(G)
6360 GOTO6370
6365 D=10-INT(G)
6370 IFD<>10THEN6376
6372 D=0
6374 C=C+1:C$=STR$(C)
6376 D$=STR$(D)
6380 EG=(G-INT(G))*3
6390 IFGA<0THEN6440
6400 IFEG>1THEN6420
6410 EG=1:GOTO6500
6420 IFEG>2THEN6435
6425 EG=2:GOTO6500
6435 EG=3:GOTO6500
6440 IFEG>1THEN6460
6450 EG=3:GOTO6500
6460 IFEG>2THEN6480
6470 EG=2:GOTO6500
6480 EG=1
6500 E$=T$(EL,EG)
6600 Q$=A$+" "+B$+C$+D$+" "+E$
6605 PRINT
6610 PRINTQ$
6615 PRINT
6620 INPUT"AUTRE TRANSFORMATION COORDONNEES->DTH (O/N) ";R$
6630 IFR$="O"THEN6050
6640 GOTO4000
```

```
1 REM------ MANIPORIC ------
2 REM
3 REM------ A T M O S ------
4 REM
100 HIMEM#9700
110 FORI=#9700TO#976A:READA$
120 IN=VAL("#"+A$):POKEI,IN:NEXT
130 CLS:PAPER0:INK2
150 PRINT:PRINT:PRINT
200 INPUT"Nombre de mots/mn (>= a 12) ";N
205 POKE#307,29
210 POKE#400,1E3/N:POKE#401,3E3/N
220 PRINT:PRINT:PRINT"  Z Pour les Points"
225 PRINT:PRINT"  V Pour les traits"
230 PRINT:PRINT"  ESPACE Pour quitter"
250 PRINTCHR$(6):CALL#9700:PRINTCHR$(6)
260 END
300 DATAA9,01,A2,00,20,90,F5,A9,00,A2,64,20,90,F5,A9,08,A2,08,20,90,F5
310 DATAA9,07,A2,FF,20,90,F5,AD,08,02,C9,38,D0,F9
320 DATA20,E8,C5,C9,20,D0,01,60,8D,02,04
330 DATAA9,07,A2,FE,20,90,F5,EA,EA,EA,EA,EA
340 DATAAD,02,04,C9,5A,D0,0C,AC,00,04,20,5F,97
350 DATA20,53,97,4C,23,97,AC,01,04,4C,44,97
360 DATAA9,07,A2,FF,20,90,F5,AC,00,04,EA,EA
370 DATAA2,C7,CA,D0,FD,88,D0,F8,AC,00,04,60
```

```
2 REM    +++++ATMORSE+++++
3 REM    +       @         +
4 REM    +BONOMO-DUTERTRE+
5 REM    + F6GKQ-F1EZH    +
6 REM    +    03-03-84     +
7 REM    +     (V.01)      +
8 REM    +    A T M O S    +
9 REM    ++++++++++++++++++
10 REM*******ATMORSE*********
20 HIMEM#8000
30 FORN=#8009TO#81BC
40 READA:POKEN,A
50 NEXT
60 FORN=#8500TO#853E
70 READA:POKEN,A
80 NEXT
81 N=#820C
82 READA
83 IFA=#FETHEN85
84 POKEN,A:N=N+1:GOTO82
85 FORN=#8600TO#865A:POKEN,0:NEXT
87 FORN=#8629TO#865A:READA:POKEN,A:NEXT:POKE#8620,1
90 PRINTCHR$(12)
95 PRINTSPC(10)
100 PRINTCHR$(4);CHR$(27);"JORICMORSE";CHR$(4)
110 PLOT5,10,"E--EMISSION"
120 PLOT5,12,"R--RECEPTION"
125 PLOT5,14,"T--ENTRAINEMENT"
130 GETA$:IFA$=""THEN130
140 IFA$="R"THEN400
150 IFA$="E"THEN500
155 IFA$="T"THEN800
160 GOTO130
400 INK7:PAPER0:CLS
405 FORN=1TO25:PRINT:NEXT
410 PRINTCHR$(4);CHR$(27);"J          RETURN POUR EMETTRE";CHR$(4)
430 CALL#8009
435 POKE#27E
436 PLOT10,14,"EMISSION"
437 WAIT300
440 GOTO500
500 REM  EMISSION
505 A$="CQ CQ CQ CQ DE F1EZH..MESSAGE DE TEST MORSE AVEC ORIC.1 A L AIDE"
510 A$=A$+" DU PROGRAMME A DENIS F6GKQ ET EDDY F1EZH.CE MESSAGE NE DOIT "
515 A$=A$+"AVOIR AU MAXIMUM QUE 254 CARACTERES.BON TRAFIC AVEC ORIC.1"
516 CLS:PRINT"LE MESSAGE EN MEMOIRE EST:":PRINT:PRINTA$
520 IFLEN(A$)>254THENL=254ELSEL=LEN(A$)
```

```
525 FORN=1TOL:POKE#82FF+N,ASC(MID$(A$,N,1)):NEXT
530 POKE#82FF+N,62
535 PRINT:PRINT:PRINT
540 INPUT"COMMUTATION(O/N)";R$
545 IFR$="O"THENPOKE#45,7ELSEPOKE#45,247
550 INPUT"VITESSE(1 a 10)";V
555 IFV>10ORV<1THEN550
560 V=(11-V)*8
570 POKE#8200,V:POKE#8201,V*3:POKE#8202,V*2
573 CLS:PAPER3
575 PLOT1,21,"----------------------------------------"
576 POKE#27C,32:POKE#27D,3:POKE#27E,21
577 GOSUB700
580 SOUND1,90,0
590 PRINTCHR$(6):CALL#820D:PRINTCHR$(6)
595 POKE#27C,16:POKE#27D,4:POKE#27E,27:CLS
600 PAPER7:GOTO110
700 PLOT1,22,"TOUCHES CURSEUR:VOLUME"
710 PLOT1,24,"TOUCHE '#'      :MESSAGE MEMORISE"
720 PLOT1,26,"TOUCHE 'RETURN':RECEPTION"
730 RETURN
800 CLS
802 PLOT3,13,"PATIENTEZ JE COMPOSE LA DICTEE"
805 FORN=0TO253
810 G=INT(RND(1)*90)
815 IFG<33THEN810
820 IFPEEK(#8600+G)=0THEN970
830 POKE#8300+N,G
840 NEXT
845 POKE#8300+N,62
847 CLS:INK0
850 INPUT"VITESSE(1 a 10)";V
855 V=(11-V)*8
860 POKE#8200,V:POKE#8202,V*2:POKE#8201,V*3
870 PRINT:PRINT"APPUYEZ SUR UNE TOUCHE POUR COMMENCER"
880 GETA$:IFA$=""THEN880
885 CLS
886 WAIT100:POKE#45,247
890 SOUND1,90,0:PAPER0:CALL#82D0
900 PAPER7
910 PLOT3,20,"D-POUR UNE AUTRE DICTEE"
920 PLOT3,22,"A-POUR ARRETER"
930 GETG$
940 IFG$="D"THEN800
950 IFG$="A"THENCLS:GOTO110
960 GOTO930
970 IFPEEK(#8300+N-1)=32THEN810
```

```
980 POKE#8300+N,32:NEXT
990 GOTO845
1000 REM***CODES MACHINE***
1010 DATA#20,#0E,#81,#A9,#00,#8D,#02,#03,#A9,#10,#8D,#00,#80,#A9
1020 DATA#15,#8D,#02,#80,#8D,#06,#80,#A9,#00,#8D,#03,#80,#8D,#04
1030 DATA#80,#8D,#05,#80,#EA,#AD,#00,#03,#29,#10,#D0,#F9,#AD,#00
1040 DATA#03,#29,#10,#F0,#F9,#98,#48,#A0,#07,#A2,#CA,#CA,#D0,#FD
1050 DATA#88,#D0,#F8,#68,#A8,#AD,#00,#03,#29,#10,#CD,#00,#80,#D0
1060 DATA#0D,#A9,#FF,#CD,#01,#80,#F0,#0E,#EE,#01,#80,#4C,#38,#80
1070 DATA#8D,#00,#80,#AD,#00,#80,#F0,#5B,#AD,#06,#80,#0A,#EA,#CD
1080 DATA#01,#80,#F0,#13,#10,#02,#4C,#82,#80,#0E,#05,#80,#0E,#04
1090 DATA#80,#A9,#00,#8D,#01,#80,#4C,#57,#80,#AD,#04,#80,#0A,#18
1100 DATA#6D,#05,#80,#AA,#BD,#00,#85,#AA,#20,#3A,#81,#AD,#00,#03
1110 DATA#29,#AF,#C9,#AF,#F0,#6B,#A9,#00,#8D,#04,#80,#8D,#05,#80
1120 DATA#AD,#06,#80,#0A,#0A,#0A,#EA,#EA,#CD,#01,#80,#F0,#08,#10
1130 DATA#03,#4C,#B8,#80,#4C,#A9,#81,#A2,#20,#20,#3A,#81,#4C,#A8
1140 DATA#81,#0E,#02,#80,#AD,#01,#90,#0A,#18,#6D,#01,#80,#CD,#02
1150 DATA#80,#F0,#25,#10,#03,#4C,#F5,#80,#AD,#01,#80,#CD,#02,#80
1160 DATA#F0,#02,#10,#05,#AD,#03,#80,#F0,#11,#A9,#01,#8D,#03,#80
1170 DATA#EE,#05,#80,#AD,#01,#80,#8D,#02,#80,#4C,#7A,#80,#AD,#01
1180 DATA#80,#8D,#06,#80,#A9,#00,#8D,#03,#80,#EE,#04,#80,#4C,#EC
1190 DATA#80,#A9,#F7,#8D,#02,#03,#58,#60,#EA,#78,#A9,#FF,#AA,#A9
1200 DATA#07,#20,#90,#F5,#A9,#00,#AA,#A9,#0E,#20,#90,#F5,#A0,#00
1210 DATA#A9,#2D,#99,#40,#BF,#C8,#98,#C9,#28,#D0,#F5,#A9,#00,#8D
1220 DATA#07,#80,#A2,#A9,#17,#8D,#7E,#02,#60,#EA,#EA,#AD,#07,#80
1230 DATA#F0,#20,#C9,#01,#F0,#32,#C9,#02,#F0,#3A,#B9,#A8,#BE,#C9
1240 DATA#2D,#F0,#3F,#EA,#EA,#EA,#EA,#EA,#8A,#99,#A8,#BE,#C8,#60
1250 DATA#A9,#00,#8D,#07,#80,#A8,#EA,#EA,#EA,#EA,#EA,#8A,#99,#A8
1260 DATA#BB,#C8,#98,#C9,#00,#F0,#01,#60,#EE,#07,#80,#A0,#00,#60
1270 DATA#EA,#EA,#EA,#EA,#EA,#8A,#99,#A8,#BC,#4C,#68,#81,#EA,#EA
1280 DATA#EA,#EA,#EA,#8A,#99,#A8,#BD,#4C,#68,#81,#8A,#48,#20,#CE
1290 DATA#CC,#A9,#FF,#AA,#A9,#07,#20,#90,#F5,#A9,#00,#AA,#A9,#0E
1300 DATA#20,#90,#F5,#68,#AA,#A9,#00,#F0,#B1,#AD,#07,#80,#F0,#03
1310 DATA#4C,#7A,#80,#98,#C9,#01,#D0,#F8,#A9,#02,#8D,#01,#80,#4C,#5A,#80
1320 REM***TABLE DE TRANSCODAGE***
1330 DATA#20,#54,#45,#4D,#4E,#41,#49,#4F,#47,#4B,#44,#57,#52,#55
1340 DATA#53,#20,#20,#51,#5A,#59,#43,#58,#42,#4A,#50,#3A,#4C,#20
1350 DATA#46,#56,#48,#30,#39,#20,#38,#20,#20,#20,#37,#20,#3E,#2E
1360 DATA#20,#2C,#2F,#2D,#36,#31,#20,#22,#3F,#20,#2A,#20,#23,#32
1370 DATA#20,#5E,#20,#33,#20,#34,#35
1380 REM***EMISSION***
1390 DATA#60,#20,#78,#EB,#C9,#00,#F0,#F9,#C9,#0D,#F0,#F4,#AA,#20
1400 DATA#A6,#82,#BD,#00,#86,#20,#27,#82,#4C,#0D,#82,#EA,#EA
1410 DATA#C9,#01,#F0,#5C,#78,#A0,#08,#8C,#03,#82,#2A,#B0,#0C,#CE
1420 DATA#02,#82,#D0,#F8,#58,#60,#EA,#EA,#EA,#EA,#EA,#CE,#03,#82
1430 DATA#F0,#39,#2A,#B0,#1F,#A2,#05,#48,#A9,#08,#20,#94,#82,#AE
1440 DATA#00,#82,#20,#75,#82,#A2,#00,#A9,#08,#20,#9D,#82,#AE,#00
```

```
1450 DATA#82,#20,#75,#82,#68,#4C,#40,#82,#A2,#05,#48,#A9,#08,#20
1460 DATA#94,#82,#AE,#01,#82,#4C,#53,#82,#A0,#FF,#88,#D0,#FD,#CA
1470 DATA#D0,#F8,#60,#AE,#01,#82,#20,#75,#82,#4C,#39,#82,#AE,#01,#82,#20
1480 DATA#75,#82,#AE,#02,#82,#20,#75,#82,#60,#20,#90,#F5,#A5,#45
1490 DATA#8D,#02,#03,#60,#20,#90,#F5,#A9,#F7,#8D,#02,#03,#60,#C9
1500 DATA#0A,#F0,#12,#C9,#0B,#D0,#1A,#AD,#49,#82,#C9,#0F,#F0,#06
1510 DATA#EE,#49,#82,#EE,#68,#82,#60,#AD,#49,#82,#F0,#06,#CE,#49
1520 DATA#82,#CE,#68,#82,#60,#C9,#23,#F0,#04,#20,#7C,#F7,#60,#A9
1530 DATA#00,#8D,#05,#82,#AD,#05,#82,#AA,#BD,#00,#82,#C9,#3E,#F0,#10
1540 DATA#AA,#20,#7C,#F7,#BD,#00,#86,#20,#27,#82,#EE,#05,#82,#4C
1550 DATA#D5,#82,#A9,#00,#A2,#00,#60,#FE,#2D,#00,#00,#38,#00,#55
1560 DATA#32,#3F,#2F,#27,#23,#21,#20,#30,#38,#3C,#3E,#00,#00,#00
1570 DATA#00,#00,#4C,#00,#05,#18,#1A,#0C,#02,#12,#0E,#10,#04,#17
1580 DATA#0D,#14,#07,#06,#0F,#16,#1D,#0A,#08,#03,#09,#11,#0B,#19
1590 DATA#1B,#1C
```

CONCLUSION

Nous avons passé en revue diverses possibilités, parfois cachées, de l'ORIC-1, ainsi que les applicaltions spécialisées dans la communication d'amateur.

Nous souhaitons avoir convaincu le lecteur de la simplicité de la programmation en langage machine ouvrant bien des horizons inaccessibles par le BASIC.

Les ressources matérielles, internes à l'ORIC-1, ont été exploitées et il peut s'avérer nécessaire de développer certains circuits d'interfaces spécifiques. Le VIA interne ne pouvant pas être utilisé à 100 % de ses possibilités, nous conseillons d'adjoindre rapidement au micro-ordinateur un VIA extérieur connecté sur le bus d'extension.

Munie d'un circuit de décodage d'adresse simplifié, cette carte pourra supporter les entrées-sorties nécessaires à :

PROGRAMMATION D'EPROM
CRAYON LUMINEUX
ACQUISITION DE SIGNAUX ANALOGIQUES,
GÉNÉRATION DE SIGNAUX ANALOGIQUES
COMMANDE DE MOTEURS ANTENNES
ETC...

Vous disposerez ainsi d'un mini-laboratoire programmable dont les applications pourront évoluer aux goûts et grés de chacun.

théoric

LA REVUE DES PASSIONNÉS D'ORIC

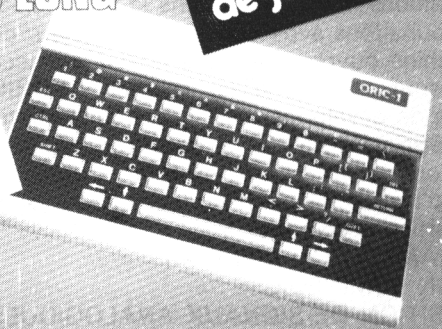
UNE VISITE CHEZ ORIC PRODUCTS INTERNATIONAL

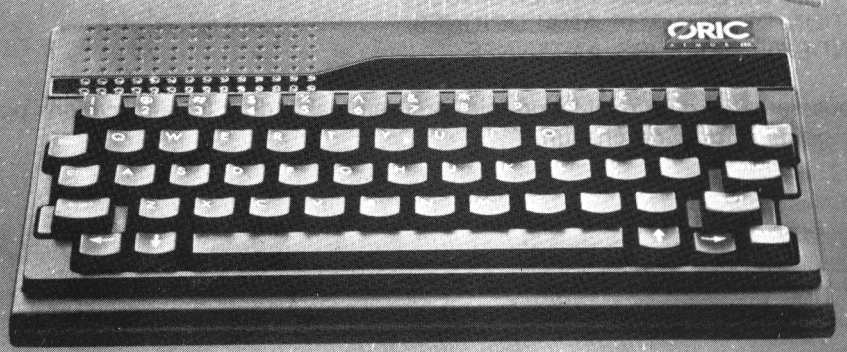
L'ATMOS AU BANC D'ESSAI

JEU: LE MOT LE PLUS LONG

tous les 2 mois chez votre marchand de journaux

GAGNEZ UN VOYAGE AUX CANARIES





BIMESTRIEL - N°1 - 20 F - MARS 1984

SOMMAIRE

PRÉSENTATION DE L'ORIC-1

INTRODUCTION AU LANGAGE

LE MICROPROCESSEUR ET L'ASSEMBLEUR

LES PROGRAMMES

ANNEXE – PROGRAMMES MODIFIÉS POUR ATMOS

QUESTIONNAIRE A PROPOS DE CE LIVRE

(à découper et à retourner aux éditions SORACOM)

1 Ce livre a-t-il répondu à votre attente ?

. . . . . .

2 Quels sont les chapitres qui vous ont semblé:
intéressants: . . . . . .
peu intéressants: . . . . . .
pas intéressants: . . . . . .

3 Avez-vous trouvé des erreurs ? Si oui, décrivez-les brièvement en indiquant la page concernée.

. . . . . .
. . . . . .
. . . . . .

4 Quels sont les sujets qui, d'après vous, méritent un développement plus complet ?

. . . . . .
. . . . . .
. . . . . .

5 Remarques personnelles.

. . . . . .
. . . . . .
. . . . . .

QUESTIONNAIRE GÉNÉRAL

(à découper et à retourner aux éditions SORACOM)

1 Dans quelles circonstances avez-vous découvert les éditions SORACOM ?

par la publicité . . . . . .
à l'occasion d'un cadeau . . . . . .
au cours d'une discussion . . . . . .
par hasard . . . . . .
à l'occasion du salon du livre . . . . . .

2 Dans quel lieu avez-vous acheté votre premier livre Soracom ?

. . . . . .

3 Quel livre a retenu le plus votre attention ?

. . . . . .

4 Ces livres répondent-ils à votre attente ?

. . . . . .

5 Souhaiteriez-vous un éventail de livres plus important ?
Si oui, dans quels domaines ? . . . . . .

. . . . . .

6 Connaissez-vous la revue MÉGAHERTZ éditée mensuellement par les éditions SORACOM ?

. . . . . .

7 Remarques personnelles

. . . . . .
. . . . . .
. . . . . .

OUVRAGES PARUS AUX ÉDITIONS SORACOM

Radio - Ondes courtes

La Guerre des Ondes
de F. Mellet et S. Faurez

Alimentations de puissance
Collection Sélection de montages

QSO en radiotéléphonie (français-anglais) *de L. Sigrand*

Interférences TV (QRM TV)
2ème édition - collection Poche

A l'écoute des radiotélétypes
2ème édition, de J.L. Fis

Technique radio pour l'amateur
3ème édition
de F. Mellet et S. Faurez

Télévisions du monde
de P. Godou

Le radioamateur et la QSL
de G. Lelarge

Technique de la B.L.U.
2ème édition, de G. Ricaud

La réception des satellites météo
de L. Kuhlmann

Les synthétiseurs de fréquences
de M. Levrel

Concevoir un émetteur expérimental *de P. Loglisci*

La propagation des ondes VHF
S. Canivenc

Avenir du futur

Transat Terre-Lune
Union pour la Promotion de la Propulsion Photonique

Aventure vécue

Expéditon
Pôle Nord Magnétique 1983
de M. Uguen

Trois p'tits mousses ...
... et puis s'en vont ...
de Bernard et Magdeleine Perret

Informatique

Programmes pour votre ORIC
de E. Jacob et J. Portelli

Communiquez avec votre ZX81
2ème édition
de D. Bonomo et E. Dutertre

Communiquez avec votre ORIC-1 et ATMOS
de D. Bonomo et E. Dutertre

Apprenez l'électronique avec ORIC-1 et ATMOS
de P. Beaufils

Interfaces pour ORIC/ATMOS
de M. Levrel

Jouez au LASER *-collection Poche*
de E. Dutertre

Les mystères du LASER 200
de D. Bourquin

Les REVUES

Mégahertz - Revue Européenne d'Ondes Courtes — *Mensuel*

Mégahertz hors-série
Spécial Informatique

Laser Info — *Trimestriel*

Théoric — *Bimestriel*

L'Hectorien — *Trimestriel*

Bretagne Éditions — *Trimestriel*

Les LOGICIELS

Canada (ORIC-1/ATMOS)
de N. Menoux

Communiquez avec votre ZX81
D. Bonomo et E. Dutertre

Las Vegas (Laser 200)
E. Dutertre

Renumber (Laser 200)
D. Bourquin

ISBN 2-904032-20-7

Photocomposé et imprimé par JOUVE - Mayenne
Maquette SORACOM
N° d'éditeur : 029
N° 12292. Dépôt légal : Mai 1984

PRIX : 145 F. TTC

COMMUNIQUEZ AVEC VOTRE ORIC 1 ET VOTRE ATMOS

Denis BONOMO Eddy DUTERTRE