LES MYSTERES D'ALICE

Alain Bonneaud

ou la pratique du 6803

SORACOM editions

© Editions SORACOM – 1984
ISBN 2- 904 032 - 30-4

INTRODUCTION

ALICE est un micro-ordinateur construit autour du microprocesseur 6803 de MOTOROLA. Ce microprocesseur gère un bus mémoire sur lequel sont connectés :

- *la ROM 8 k octets de Basic Microsoft*
- *2 RAM de 2 k octets chacune*
- *le circuit MC 6847 de MOTOROLA, qui est un générateur contrôleur d'écran*
- *le bus d'extension*
- *le clavier*

La ROM Basic est implantée entre les adresses E000 et FFFF en hexadécimal, soit les 8 derniers k octets adressables par le microprocesseur.

La RAM est constituée de deux circuits intégrés 4016 de 2 k octets chacun. L'espace RAM adressable dans la version de base va des adresses 4000 à 5000 en hexadécimal. L'extension mémoire de 16 k se connecte sur le bus d'extension et permet d'étendre l'espace adressable jusqu'à l'adresse 9000 en hexadécimal.

Le circuit MC 6847 gère l'affichage sur l'écran. Il possède une ROM intégrée contenant le générateur de caractères.

Le bus adresse et donnée multiplexés arrive au connecteur d'extension. En effet, une particularité du MC 6803 est d'avoir les 8 bits de poids faible du bus adresse multiplexés avec le bus de données. Le signal AS du MC 6803 sert à échantillonner les adresses pour le démultiplexage.

Le clavier est connecté d'une part sur le port 1 du 6803 et d'autre part sur le bus de données et le port 2. Chaque touche du clavier met en contact un fil du bus de données avec un fil du port 1.

ALICE est alimentée par un transformateur extérieur qui délivre du 12 volts alternatifs. Les diverses tensions continues dont a besoin le micro-ordinateur sont fabriquées sur la carte mère elle-même. Soit du 5 V, du + 12 V, et du − 12 V.

L'oscillateur qui fabrique l'horloge de base est intégré dans le MC 6803. Un quartz extérieur permet de sélectionner la fréquence. Sur ALICE, cette fréquence est de 3,559 MHz.

Dans la première partie de cet ouvrage, nous nous attacherons à donner une description technique la plus complète possible de cette merveilleuse petite machine pour l'apprentissage du Basic que constitue ALICE. L'étude que nous avons menée, en particulier sur les possibilités du système nous a conduit à penser qu'il était possible d'aller beaucoup plus loin que l'apprentissage du Basic sur ce micro-ordinateur et, en particulier, nous avons trouvé des possiblilités de programmation en language machine qui ne sont pas décrites dans le manuel de base, ainsi que quelques instructions Basic supplémentaires qu'il est possible d'utiliser sans aucune modification sur la version de base.

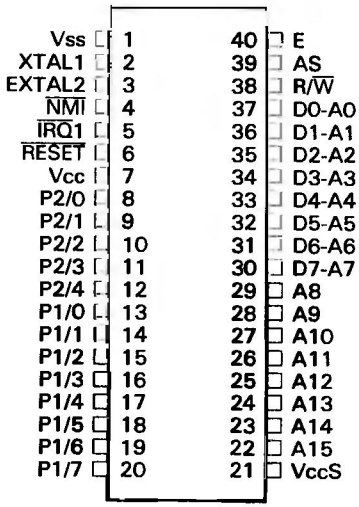
Dans la seconde partie, nous donnerons quelques exemples de programmes réalisés sur ALICE, et recouvrant divers domaines : jeux, utilitaires, gestion familiale, EAO, etc... Certains de ces programmes peuvent fonctionner sur la version de base, d'autres, plus volumineux, nécessiteront l'extension mémoire. Tous les programmes que nous avons réalisés sur plus de 3 k l'ont été avec l'extension 16 k de TANDY.

LE MICROPROCESSEUR MC 6803

Le microprocesseur MC 6803P, autour duquel sont réalisés le micro-ordinateur ALICE de Matra- Hachette et le micro-ordinateur MC10 de Tandy est un microprocesseur 8 bits qui emploie un système de multiplexage des adresses et des données, permettant d'étendre la capacité mémoire adressable à 64 k octets. Il sagit en fait d'un microprocesseur récent et encore peu utilisé sur les micro-ordinateurs, dérivé du 6800 de Motorola dont le code objet est entièrement compatible avec le 6803. Cependant, par rapport au 6800, le MC 6803 présente un certain nombre d'extensions. Le language machine du 6803 comporte en effet 10 nouvelles instructions sur 16 bits dont, en particulier une multiplication réalisée par hardware sur deux données de huit bits, résultat sur 16 bits. Nous étudierons en fin de ce chapitre l'intégralité du jeu d'instructions du 6803.
De plus ce microprocesseur intègre une RAM interne de 128 octets (de l'adresse '080 à l'adresse 'OFF), une horloge interne, un circuit d'interface série, un système d'entrées/sorties parallèles et un temporisateur programmable sur 16 bits. Le circuit de division par quatre de l'horloge interne autorise l'utilisation d'un quartz 3,58 MHz, plus économique qu'un quartz 1 MHz. L'alimentation unique se fait en +5 volts, ce qui rend ce circuit compatible TTL. Par ailleurs, il est possible de prévoir une alimentation de secours qui, en cas de coupure de l'alimentation principale permettra de sauvegarder le contenu des 32 premiers octets de la RAM interne, qui disposent d'un mode à faible consommation (+5 V/8 mA) et contiennent des informations indispensables au fonctionnement correct du micro-ordinateur. Extérieurement, le 6803P se présente sous la forme d'un circuit CMOS, enfermé dans un boîtier plastique (suffixe P) de 40 broches dont le schéma est le suivant :

***NOTA** : par convention, dans la suite de cet ouvrage, lorsqu'un nombre sera précédé du symbole ' , cela signifiera que ce nombre est écrit ne hexadécimal.*
gardés en mode faible consommation avec un courant de 8 mA maximum. Il sera alors

BROCHAGE DU 6803
(vu de dessus)



Ref. des broches	Fonctions
Vss	masse (0 volts)
XTAL1	broches du quartz
EXTAL2	(4 × fréq. horloge interne)
NMI	demande d'interruption non masquable
IRQ1	demande d'interruption
RESET	réinitialisation
Vcc	alimentation (+5 V ±5%)
P2/0 — P2/4	E/S parallèles port 2
P1/0 — P1/7	bus E/S parallèles port 1
E	validation de l'horloge
AS	échantillonnage des adresses
R/W	lecture/écriture
D0-A0 — D7-A7	bus adressas et données multiplexées
VccS	alim. da sauvegarde (+5 V)

Le 6803P peut donc être décomposé en neuf éléments fonctionnels

le CPU
une RAM interne de 128 octets
un multiplexeur
un bus multiplexé données/adresses
un bus adresses
le temporisateur
le circuit d'interface (ACIA)
le Port 2 d'E/S, bidirectionnel
le Port 1 d'E/S, bidirectionnel

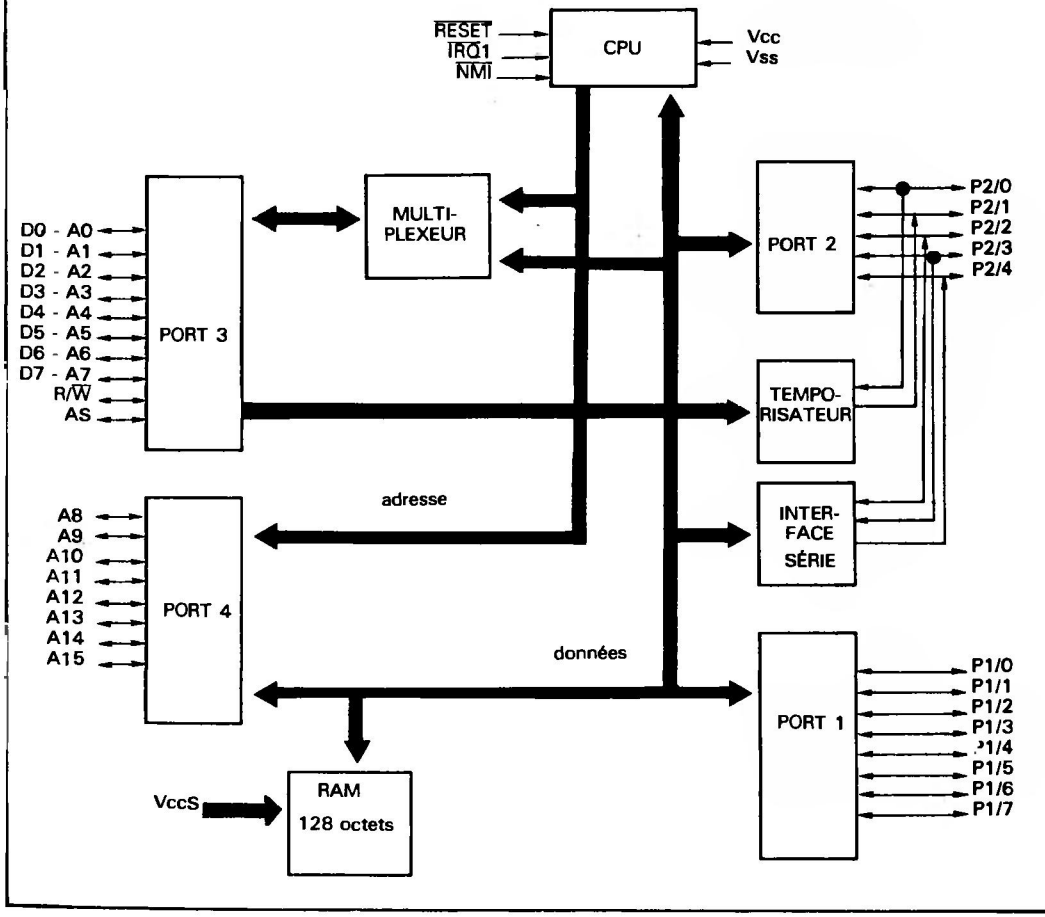
DÉFINITION DES SIGNAUX

La définition succinte des signaux a été donnée dans le schéma présentant le brochage du 6803. Nous allons maintenant la reprendre pour chaque signal appliqué à une broche du boîtier, d'une façon beaucoup plus détaillée.

• ***Vcc et Vss*** (broches 7 et 1) : elles sont utilisées pour fournir au microprocesseur, respectivement une alimentation de +5 volts (±5%) et une masse.

• ***XTAL1 et EXTAL2*** (broches 2 et 3) : il sagit de deux broches réservées à un quartz. Un circuit diviseur par quatre est intégré au boîtier ainsi, d'ailleurs, que l'oscillateur qui fabrique l'horloge de base, ce qui permet d'utiliser un quartz de 4 MHz pour faire une horloge interne à 1 MHz. EXTAL2 peut être reliée à une horloge externe mais dans ce cas, il est nécessaire de mettre XTAL1 à la masse.

Nous obtenons donc **le schéma fonctionnel** suivant :



• ***VccS*** (broche 21) : cette broche est destinée à fournir une tension de sauvegarde à la RAM interne du 6803 pendant une coupure d'alimentation. Cette broche devra alors être reliée à une batterie de secours délivrant du +5 V. En cas de coupure de l'alimentation générale, les 32 premiers octets de la RAM interne, qui contiennent des informations indispensables au 6803 et que nous étudierons dans les pages suivantes, pourront être sauvegardés en mode faible consommation avec un courant de 8 mA maximum. Il sera alors nécessaire de mettre à 0 le bit de validation de la RAM dans le registre de contrôle de la mémoire interne (bit 6 de l'adresse '0014).

• ***$\overline{RESET}$*** (broche 6) : cette broche est destinée à une réinitialisation du microprocesseur soit après la mise sous tension, soit après une coupure de l'alimentation générale. On peut remarquer que les broches $\overline{RESET}$, $\overline{IRQ1}$, $\overline{NMI}$ (voir schéma du brochage), dont le nom est surmonté d'une barre, fonctionnent en logique négative, c'est-à-dire que la validation est effective, à l'inverse de tous les autres signaux, pour un niveau bas (c'est-à-dire un 0 logique). Pour être validé, ce niveau bas sur la broche $\overline{RESET}$ doit être maintenu pendant au moins 20 ms ; c'est ce qui se passe à la mise sous tension par exemple. Ensuite, dès qu'un niveau haut est détecté sur cette entrée, le micropocesseur commence sa séquence d'initialisation en lisant le contenu des deux derniers octets adressables (adresses 'FFFE et 'FFFF) qui contiennent l'adresse de début de la séquence d'instructions permettant l'initialisation du système (sur ALICE, ce vecteur de RESTART - ou vecteur de redémarrage - contient la valeur 'F72E). A ce moment, le bit I du registre d'état (masque d'interruption) est positionné à 1 et les deux octets du vecteur de redémarrage sont placés dans le compteur ordinal qui commence l'exécution de la séquence d'inititalisation (démarrage à froid). Le bit I devra ensuite être remis à 0 pour que le microprocesseur puisse prendre en compte des interruptions masquables en provenance de périphériques (clavier par exemple...).

• ***E - Enable*** - (broche 40) : cette broche de sortie valide l'horloge externe pour le reste du système, lorsque l'oscillateur interne est utilisé. Il s'agit alors d'une horloge à une seule phase, compatible TTL dont la fréquence sera le résultat de la division par quatre de la fréquence du quartz.

• ***$\overline{NMI}$ - interruption non masquable*** - (broche 4) : cette broche est utilisée pour prévenir le MPU qu'une interruption non masquable est demandée par le système. De même que lorsqu'il s'agit d'une demande d'interruption masquable, le microprocesseur termine d'abord l'exécution de l'instruction en cours avant de prendre en compte la demande d'interruption. Le masque d'interruption (bit I du registre d'état) est sans effet sur NMI. Lors de la prise en compte d'une demande d'interruption non masquable, le registre d'index, le compteur ordinal, les deux accumulateurs ainsi que le registre d'état sont sauvegardés dans la pile. A la suite de cette opération de sauvegarde, le microprocesseur lit le contenu des adresses 'FFFC et 'FFFD qui contiennent le vecteur de branchement à la routine de traitement NMI. Ce vecteur est chargé dans le compteur ordinal et l'exécution de la routine commence. Sur ALICE le vecteur de branchement NMI, placé aux adresses 'FFFC et 'FFFD, a pour valeur '4212.
Les entrées $\overline{NMI}$ et $\overline{IRQ1}$ sont des lignes d'interruption hardware qui sont échantillonnées pendant que E est à un niveau haut. La routine d'interruption débutera sur le niveau bas de la ligne E suivant immédiatement la fin d'exécution de l'instruction en cours au moment de l'échantillonnage.

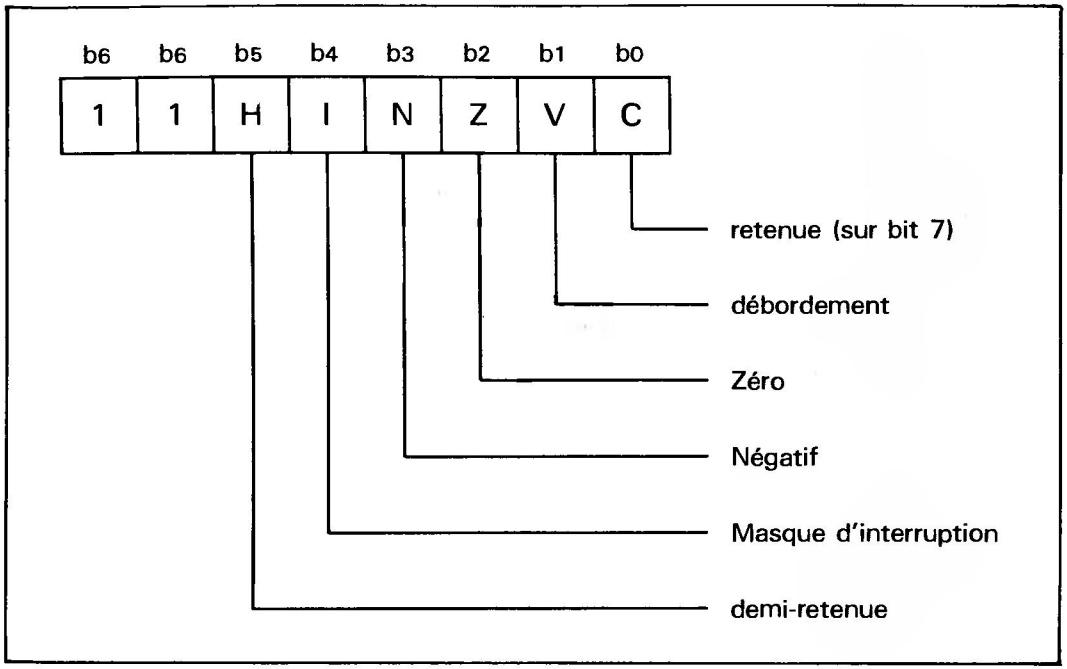
• ***$\overline{IRQ1}$ - interruption masquable*** - (broche 5) : il s'agit encore d'une broche d'entrée, fonctionnant (comme $\overline{NMI}$) en logique négative. Un $\emptyset$ sur cette broche prévient le microprocesseur qu'une demande d'interruption a été générée par un organe du micro-ordinateur. Le microprocesseur termine d'abord l'exécution de l'instruction en cours avant de prendre en compte cette demande. Sur le niveau bas de la ligne E suivant immédiatement la fin de cette instruction, il va alors tester le masque d'interruption (bit I du registre d'état). Si le masque d'interruption est positionné à 1, il ne prendra pas en compte la demande et poursuivra normalement l'exécution de son programme.
Si le masque d'interruption est positionné à $\emptyset$, alors le microprocesseur va d'abord empi-

ler le registre d'index, le compteur ordinal, les deux accumulateurs et le registre d'état de sauvegarder leur contenu et d'effectuer un retour du sous-programme d'interruption dans de bonnes conditions permettant au programme de continuer à s'exécuter. Ensuite le MPU positionne le masque d'interruption à 1, de façon à inhiber toute nouvelle demande d'interruption masquable pendant l'exécution de la routine IRQ1, et il va lire aux adresses 'FFF8 et 'FFF9 la valeur du vecteur de branchement à la routine (sur ALICE cette valeur est '42ØC) avant de commencer l'exécution de cette routine. $\overline{IRQ1}$ est utilisée par des organes extérieurs au 6803. Mais il existe également la possiblité pour des éléments internes au boîtier (par exemple le temporisateur) de formuler une demande d'interruption masquable. Dans cas ils utilisent une ligne interne d'interruption : IRQ2. Toute demande d'interruption interne IRQ2 sera gérée par le MPU d'une façon identique à IRQ1 avec cependant des vecteurs de branchement aux routines de traitement d'interruption différents. Il existe des niveaux de priorité différents selon le type d'interruption : aussi, si à un même instant t deux demandes $\overline{IRQ1}$ et IRQ2 d'interruptions sont formulées simultanément, $\overline{IRQ1}$ ayant un niveau de priorité plus élevé sera prise en compte la première. Notons que le masque (bit I) d'interruption permet d'inhiber, lorsqu'il est positionné à 1, à la fois les interruption IRQ1 et IRQ2.
Nous allons maintenant donner sous forme d'un tableau, un résumé des données concernant ces différents types d'interruptions. L'ordre du tableau correspond à des priorités décroissantes (priorité la plus élevée pour le redémarrage (RESTART) et priorité la plus faible pour l'interruption IRQ2 sur entrée/sortie série).

type d'IT	description	adresse du vecteur de branchement		valeur hexa du vecteur de branchement sur ALICE
		hexa	décimale	
$\overline{RESET}$	démarrage à froid du système	FFFE-FFFF	65534-65535	F72E
NMI	interruption non masquable	FFFC-FFFD	65532-65533	4212
SWI	interruption programmée	FFFA-FFFB	65530-65531	420F
IRQ1	interruption masquable externe	FFF8-FFF9	65528-65529	420C
IRQ2	interruption registre entrée du timer	FFF6-FFF7	65526-65527	4209
IRQ2	interruption registre sortie du temporisateur	FFF4-FFF5	65524-65525	4206
IRQ2	débordement temporisateur	FFF2-FFF3	65522-65523	4203
IRQ2	interruption de l'interface série	FFF0-FFF1	65520-65521	4200

On voit, sur le schéma du registre d'état 6803, que le bit I qui constitue le masque d'interruption est le bit 4 du registre d'état.

CONFIGURATION DU REGISTRE D'ÉTAT DU 6803

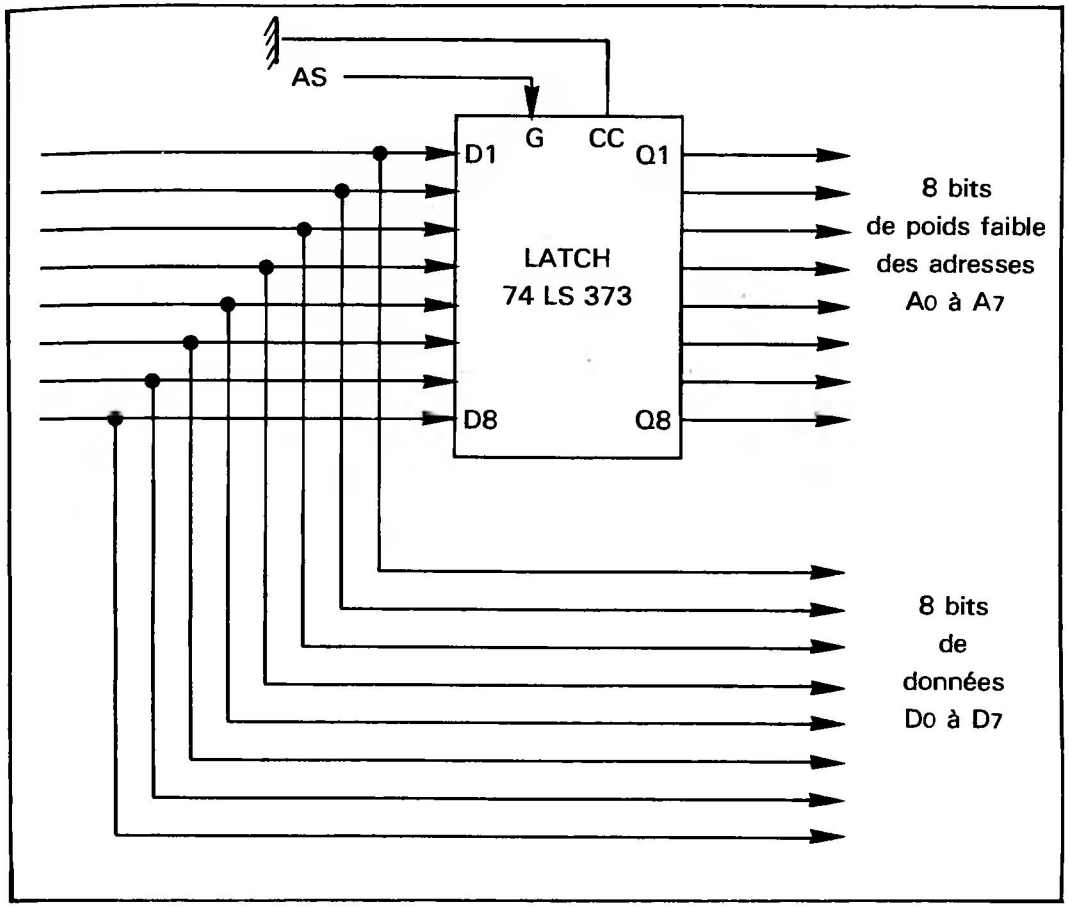


• ***AS - Adress Strobe*** - (broche 39) : il s'agit d'une sortie qui sert à échantillonner les adresses pour le démultiplexage. En effet, les huit bits de poids faible des adresses sont multiplexés sur le bus de données. Cette broche de sortie sert à envoyer un signal à un latch 8-bits qui se trouve sur la carte mère d'ALICE : le 74 LS 373. Lorsque le signal est envoyé par AS, le latch verrouille les huit bits de poids faible des adresses et ainsi ce sont des données qui sont envoyées sur le bus pendant que E envoie une impulsion. Le schéma suivant montre comment est connecté le 74 LS 373 qui permet cet échantillonnage conjointement avec Adress Strobe.

• ***R/$\overline{W}$ - Read/$\overline{Write}$*** - (broche 38) : ce signal de sortie permet d'indiquer aux circuits mémoires et périphériques que le microprocesseur est dans un état "Lecture" si la ligne R/$\overline{W}$ est dans un état haut (R/$\overline{W}$ = 1) ou dans un état d'écriture si le signal envoyé sur la ligne R/$\overline{W}$ est dans un état bas (R/$\overline{W}$ = 0). Au repos, l'état normal de ce signal est un état haut (lecture).

• ***bus adresses/données multiplexées A0-D0 à A7-D7*** (broches 30 à 37) : huit broches sont utilisées par ce bus multiplexé. Sur ces huit broches on trouve à la fois les huit bits de poids faible du bus d'adresses et les huit bits du bus de données bidirectionnel. Un latch externe peut être utilisé comme c'est le cas sur ALICE avec le 74 LS 373 pour atteindre l'intégralité des 16 bits du bus d'adresses, sous le contrôle du signal AS.

• ***bus d'adresses A8-A15*** (broches 22 à 29) : ces huit broches fournissent les huit bits de poids fort du bus d'adresses, permettant ainsi d'adresser 64 k octets de mémoire.



Il nous reste enfin les broches 8 à 20 qui sont réservées aux échanges avec les périphériques. Le 6803 est en effet doté de deux ports réservés à ces échanges avec l'extérieur. Le port 1 est un port de 8 bits et le port 2 est un port de 5 bits. Tous deux peuvent être considérés comme bidirectionnels dans la mesure où chaque bit peut être configuré en entrée ou en sortie par l'intermédiaire de deux registres de direction, l'un associé au port 1 et l'autre au port 2. Ainsi un "1" dans un bit (le bit 5 par exemple) du registre de direction configurera le bit correspondant (le bit 5 dans notre exemple) du port d'E/S en SORTIE. Un "0" dans le même bit du registre de direction l'aurait configuré en ENTRÉE. On voit ainsi que chaque bit des ports d'Entrée/Sortie peut être configuré en entrée ou en sortie, individuellement, ceci d'une manière tout à fait indépendante des autres bits à l'intérieur d'un même port. Ainsi dans le port 1, il est possible, par exemple, d'avoir 3 bits configurés en sortie alors que les 5 autres sont configurés en entrée. Il est bien sûr possible d'adresser par programme les ports 1 et 2 ainsi que leur registre de direction associé et d'en modifier les contenus :

PORT D'E/S	ADRESSE DU REGISTRE DE DIRECTION	ADRESSE DU PORT
1	'0000	'0002
2	'0001	'0003

Vous pourrez ainsi vérifier simplement qu'en plaçant certaines valeurs à l'adresse '0000, qui est l'adresse du registre de direction du port 1 sur lequel est connecté le clavier, il vous sera possible d'interdire l'utilisation de ce dernier qui se trouve ainsi déconnecté (sur ALICE).

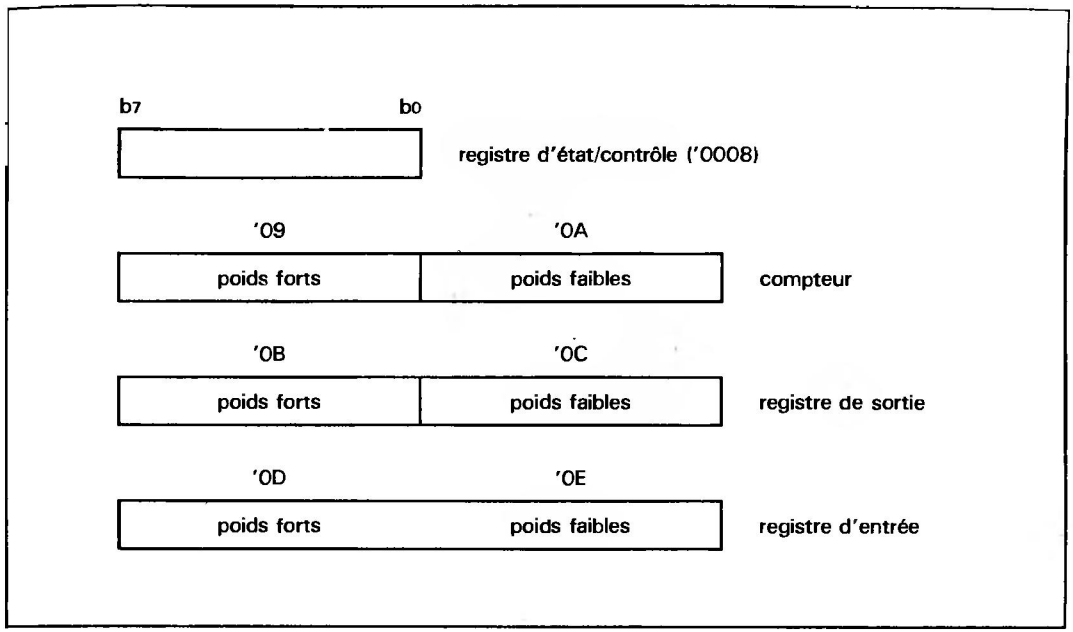
• ***Port d'E/S n° 1*** (broches 13 à 20) : il s'agit donc d'un port 8 bits sur lequel chaque bit peut être configuré en mode entrée ou en mode sortie grâce à son bit de direction associé dans le registre de direction du port 1. En entrée, la tension à envoyer sur ces broches doit être supérieure à 2,0 V pour être interprétée comme un "1" logique et inférieure à 0,8 V pour un "0" logique.
Après un redémarrage (RESET), ces ports seront configurés en entrée (0 dans les registres de direction associés).

• ***Port d'Entrée/Sortie n° 2*** : il s'agit du port 5 bits pour lequel chaque bit peut également être programmé comme entrée ou sortie grâce au registre de direction. Le bit 1 du port 2 peut servir soit en entrée, soit en sortie temporisateur. Le port 2 permet également l'accès à l'interface de communication série et au temporisateur. Le temporisateur est associé à deux lignes, l'une d'entrée "Timer input" (P2/0) et l'autre de sortie "Timer output" (P2/1). L'interface série est associée à trois lignes : transmission (P2/4), réception (P2/3) et l'horloge (P2/2). Chacun, du temporisateur et de l'interface série possède son propre registre de contrôle qui leur permet d'accéder au port d'Entrée/Sortie n° 2.

LE TEMPORISATEUR PROGRAMMABLE

Le 6803 contient un temporisateur programmable qui peut être utilisé pour mesurer des délais sur un signal d'entrée et en même temps générer une onde de sortie. La longueur des impulsions, tant en entrée qu'en sortie, peut varier de quelques micro-secondes à quelques secondes. La structure hardware du temporisateur est fondée sur :
- un registre d'état et de contrôle, sur 8 bits (à l'adresse '0008)
- un compteur temps réel sur 16 bits (adresse '0009 - '000A)
- un registre sortie sur 16 bits (adresse '000B - '000C)
- un registre entrée sur 16 bits (adresse '000D - '000E)

Ce qui nous donne le schéma suivant :



Nous allons brièvement étudier chacun de ces composants.

• ***Le compteur temps réel*** ('09:0A) : c'est l'élément clé du temporisateur. Il s'agit d'un compteur sur 16 bits qui utilise l'horloge du système. Il est donc incrémenté à chaque "top" d'horloge du 6803. La valeur contenue dans ce compteur peut être lue par le processeur à tout instant. Le compteur est réinitialisé à 0 lors d'un $\overline{\text{RESET}}$ et servira à compter des impulsions de façon à générer des délais. Lorsqu'un débordement se produira, une interruption sera générée. Le compteur peut être considéré comme un registre que l'on ne peut que lire, avec cependant une exception pour l'écriture dans le compteur par le processeur d'une seule et même valeur. En effet, toute tentative d'écriture dans ce compteur provoquera la mise à la valeur 'FFF8, et ceci quelque soit la valeur que l'on voudrait écrire.

• ***Le registre de sortie*** ('0B:0C) : il s'agit d'un registre de comptage sur 16 bits, dans lequel on peut lire ou écrire. Il sert à contrôler une sortie. Le contenu de ce registre sera constamment comparé au contenu du compteur. Or, le compteur s'incrémentant sans cesse, ceci permettra de mesurer un certain nombre d'impulsions avant qu'il n'y ait égalité entre les deux contenus. Dès que l'on trouve les deux valeurs égales, un indicateur d'état : (bit OCF du registre de contrôle) est positionné à 1 et la valeur courante du bit OLVL (Output Level) du registre de contrôle est envoyé sur le port 2. Ainsi, si le registre de direction du port 2 est tel que le bit 1 est à "1" (mode sortie) alors la valeur apparaîtra sur la broche correspondant au bit 1 du port 2. Le registre de sortie est positionné à 'FFFF par un $\overline{\text{RESET}}$.

• ***Le registre d'entrée*** ('0D:0E) : il s'agit encore d'un registre de 16 bits utilisable seulement en lecture. On y stockera la valeur courante du compteur lorsqu'une modulation cor-

recte apparaîtra au niveau du signal d'entrée. Ce changement de transition en entrée, nécessaire au déclanchement de transfert du contenu du compteur est contrôlé par le bit IEDG (Input Edge) du registre de contrôle. Le bit 0 du registre de direction du port 2 devra préalablement avoir été positionné à "0" (mode lecture) de façon à permettre à l'impulsion d'entrée d'arriver jusqu'au bit IEDG du temporisateur.

• ***Le registre d'état et de contrôle*** ('08) : il s'agit d'un registre de 8 bits dont tous les bits peuvent être lus, mais on ne pourra modifier que les 5 bits de poids faibles c'est-à-dire b0 à b4. Les 3 bits restant contiendront des informations sur l'état du temporisateur :
- le bit 5 contiendra un indicateur qui sera positionné automatiquement à "1" lorsque le compteur passera en débordement de 'FFFF à '0000.
- le bit 6, dont nous avons déjà parlé, (OCF) contiendra un indicateur qui sera positonné à "1" dès qu'il y aura égalité entre le contenu du compteur et le contenu du registre de sortie.
- le bit 7 indiquera qu'une transition correcte s'est produite sur la broche d'entrée, entrainant la recopie, dans le registre d'entrée, du contenu du compteur.

Chacun de ces indicateurs peut être utilisé pour générer une interruption sur la ligne IRQ2. A chacun de ces indicateurs est associé, dans ce même registre de contrôle, un bit de validation de la demande d'interruption. Selon que le bit de validation sera positionné à "1" ou à "0", une impulsion sera ou ne sera pas envoyée sur IRQ2.
Si, le bit I (masque d'interruption) du 6803 est à "0" une interruption, dont le vecteur d'adresse et la priorité ont été précédemment définis, pourra alors être générée.

DESCRIPTION DU REGISTRE D'ÉTAT ET DE CONTRÔLE DU TEMPORISATEUR

ICF	OCF	TOF	EICI	EOCI	ETOI	IEDG	OLVL
b7	b6	b5	b4	b3	b2	b1	b0

registre d'état/contrôle (adresse '0B)

• ***bit 0 - OLVL*** (Output Level) : lorsque le contenu du registre de sortie et le contenu du compteur sont égaux, et sous réserve que le bit 0 du registre de direction du port 2 soit positionné à "0" (mode écriture), la valeur de OLVL est envoyée sur la broche de sortie.

• ***bit 1 - IEDG*** (Input Edge) : ce bit contrôle quelle transition d'entrée déclenchera la recopie du contenu du compteur dans le registre d'entrée, sous réserve que le bit 0 du registre de direction du port 2 soit positionné à "0" (mode écriture). Si IEDG est positionné à 0, alors le transfert sera réalisé sur le front descendant ; par contre, si IEDG est positionné à 1, le transfert sera réalisé sur le front montant.

• ***bit 2 - ETOI*** (Enable Timer Overflow Interrupt) : lorsque ce bit est positionné à 1, une demande d'interruption sera générée sur le bus interne (IRQ2) dès que le bit TOF passera à 1. Si ETOI est positionné à 0, aucune demande d'interruption sur TOF ne pourra être générée.

• ***bit 3 - EOCI*** (Enable Output Compare Interrupt) : lorsque ce bit est positionné à 1, une demande d'interruption sera générée sur le bus interne (IRQ2) dès que le bit OCF passera à 1. Si EOCI est positionné à 0, aucune demande d'interruption sur OCF ne sera générée.

• ***bit 4 - EICI*** (Enable Input Capture Interrupt) : lorsque ce bit est positionné à 1, une demande d'interruption sera générée sur le bus interne (IRQ2) dès que le bit ICF passera à 1. Si ETCI est positionné à 0, aucune demande d'interruption sur ICF ne sera générée.

• ***bit 5 - TOF*** (Timer Overflow Flag) : cet indicateur est positionné à 1 lorsque le compteur passe en débordement de 'FFFF à '0000. Il est remis à 0 par la seule lecture du registre de contrôle suivie de la lecture par le processeur du contenu du compteur (adresse '09).

• ***bit 6 - OCF*** (Output Compare Flag) : cet indicateur est positionné à 1 lorsqu'il y a égalité entre le registre de sortie et le contenu du compteur. Il est réinitialisé à 0 par la seule lecture du registre de contrôle suivie d'une écriture par le processeur dans le registre de sortie (adresse '0B:0C).

• ***bit 7 - ICF*** (Input Capture Flag) : cet indicateur est positionné à 1 par une transition correcte en entrée vers IEDG. Il est réinitialisé par la lecture du registre de contrôle suivie de la lecture par le processeur du registre d'entrée (adresse '0D:0E).

L'INTERFACE DE COMMUNICATION SÉRIE

Le MC 6803 est doté d'une interface série de communication full-duplex en mode asynchrone. Deux formats sont offerts par le système (NRZ St/art-Stop ou bi-phasé) à plusieurs vitesses de transmission. Le contrôleur comprend un "transmetteur" et un "récepteur" qui travaillent indépendamment l'un de l'autre mais obligatoirement en utilisant le même format pour les données et la même vitesse de transfert. Ils communiquent tous deux avec le MPU via le bus de données et avec l'extérieur par l'intermédiaire des bits 2, 3 et 4 du port d'E/S n°2.

PROCESSUS D'ACTIVATION.

Dans une application sur multi-processeur, le protocole d'échange contient normalement une adresse de destination dans les premiers octects du message. Afin de premettre au MPU's non sélectionné d'ignorer le reste du message, un processus d'activation est inclus, qui peut inhiber les interruptions ultérieures sur l'interface série jusqu'à ce que la ligne soit libérée. Le bit d'"activation" est automatiquement remis à 0 par une chaîne de dix "1" consécutifs qui indique que la ligne est libre. Le logiciel doit prévoir une courte période pendant laquelle la ligne est au repos entre deux messages consécutifs, mais il doit faire en sorte qu'aucun temps mort ne se produise à l'intérieur-même du message.

L'interface série présente un certain nombre d'options, exigées par de telles communications séries, et qu'il est possible de programmer.

OPTIONS PROGRAMMABLES

Sont programmables les options suivantes:
- le format des données (standard Start-Stop ou bi-phase)
- l'horloge sélectionnée (interne ou extérieure)
- la vitesse de transmission
- la sélection (ou non) de la procédure d'activation
- les demandes d'interruptions (autorisées individuellement pour le registre d'émission et le registre de réception)
- la sortie horloge (horloge interne sélectionnée ou non sur le bit 2 du port 2)
- les bits 3 et 4 du port 2 réservés (ou non) individuellement aux E/S séries, en mode émission ou en mode réception.

STRUCTURE DE L'INTERFACE SÉRIE

L'interface série contient quatre registres programmables dont deux en écriture seule et un en lecture seule.
- un registre d'état et de contrôle sur 8 bits
- un registre de contrôle de la vitesse et du mode de transmission sur 4 bits (en écriture seule)
- un registre de réception qui reçoit le mot à lire par le processeur, sur 8 bits (en lecture seule)
- un registre de transmission qui reçoit de l'unité centrale, le mot à émettre sur 8 bits (en écriture seule).

De plus, l'interface série utilise le bit 3 (entrée série) et le bit 4 (sortie série) du port d'E/S n° 2, le bit 2 du port 2 étant, quant à lui, utilisé si l'option horloge interne-out ou horloge externe-in est sélectionnée.

Nous allons maintenant étudier chacun de ces registres avant de voir le processus standard de réception et de transmission.

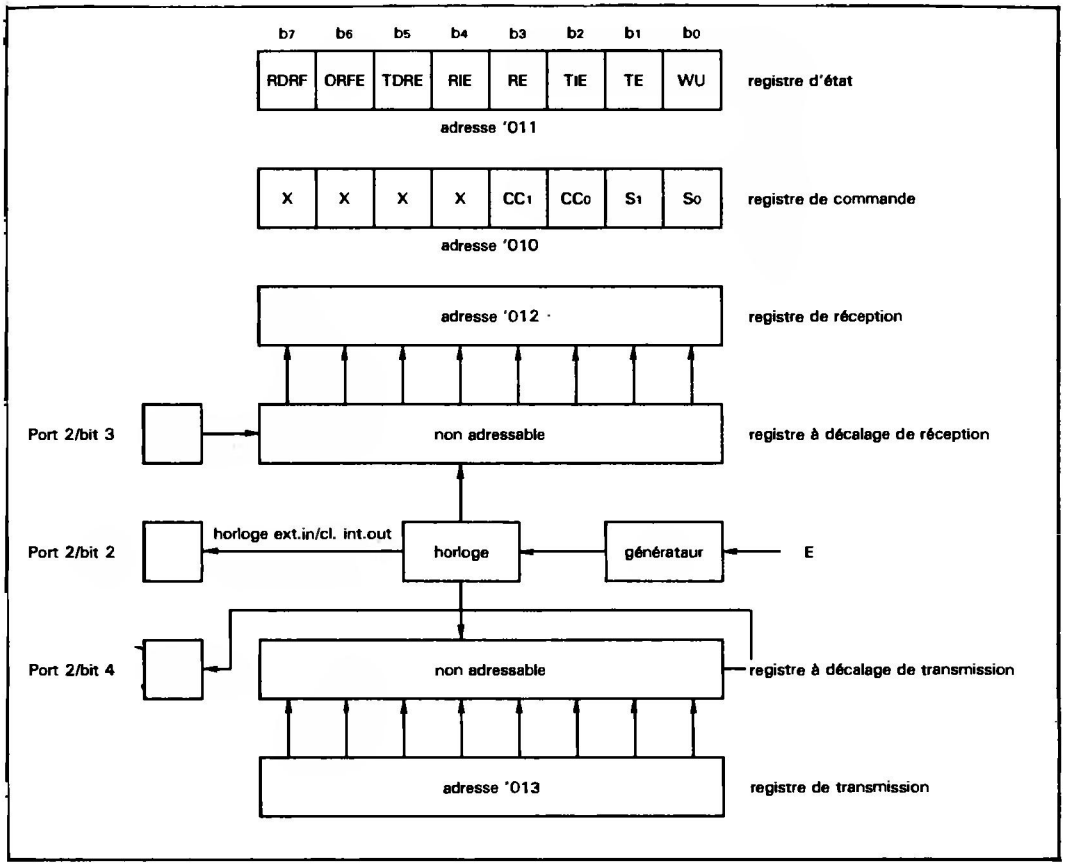
LE REGISTRE D'ÉTAT ET DE CONTRÔLE

Il s'agit d'un mot de 8 bits dans lequel les bits 5 à 7, qui sont des indicateurs d'état, sont à lecture seule alors que les bits 0 à 4 peuvent être lus ou modifiés par programme. Sur RESET, le contenu de ce registre est initialisé à '20, c'est-à-dire que le bit 5 est positionné à 1 et tous les autres à 0.

RDRF	ORFE	RIE	TDRE	RE	TIE	TE	WU
b7	b6	b5	b4	b3	b2	b1	b0

registre d'état et de contrôle (adresse '011)

SCHÉMA LOGIQUE DE L'INTERFACE SÉRIE



• ***bit 0*** - ***WU*** (Wake Up on idle line) : lorsque ce bit est positionné à 1, mise en action du processus d'activation. Ce bit sera remis à Ø automatiquement par hardware à la réception de dix "1" consécutifs.

• ***bit 1*** - ***TE*** (Transmit Enable) : lorsqu'il est positionné à "1", provoque la mise à "1" du bit 4 du registre de direction du port 2, lequel ne pourra plus être remis à "0" tant que TE restera à "1". Le passage de TE à la valeur "0" n'entraine pas automatiquement le passage à 0 du bit 4 du registre de direction du port 2. un train de neuf "1" consécutifs est généré lorsque TE passe de la valeur "0" à la valeur "1". Pendant qu'il est positionné à "1", la sortie transmetteur est envoyée sur le bit 4 du port 2.

• ***bit 3*** - ***TIE*** (Transmit Interrupt Enable) : lorsque ce bit est positionné à 1, il autorise l'envoi d'une demande d'interruption sur IRQ2 lorsque TDRE passe à "1". Lorsque TIE est à 0, il inhibe l'interruption.

• ***bit 3 - RE*** (Receiver Enable) : lorsqu'il est positionné à 1, provoque la mise à 0 du bit 3 du registre de direction du port 2, lequel ne pourra plus être remis à 1 tant que RE restera à 1. Le passage de RE à 0 n'entraine pas automatiquement le passage à 1 du bit 3 du registre de direction du port 2. Pendant que RE est positionné à 1, le bit 3 du port 2 sera envoyé vers le récepteur.

• ***bit 4 - RIE*** (Receiver Interrupt Enable) : lorsque ce bit est positionné à 1, les interruptions du récepteur sont autorisées et IRQ2 reproduit l'état de (RDRF ⊕ ORFE). Lorsque RIE est à 0, il inhibe l'interruption.

• ***bit 5 - TDRE*** (Transmit Data Registrer Empty) : positionné à 1 par hardware lorsque le contenu du registre de transmission est envoyé dans le registre à décalage correspondant (ce qui revient à "vider" le registre de transmission). Ce transfert est synchronisé sur l'horloge. TDRE est remis à 0 par une lecture du registre d'état suivie de l'écriture du caractère suivant dans le registre de transmission. Aucune donnée ne peut être transférée dans le registre de transmission tant que celui-ci n'est pas vidé, c'est-à-dire tant que TDRE n'est pas à 1. On à vu que lors d'une initialisation du système, TDRE est initialisé à 1.

• ***bit 6 - ORFE*** (Over Run - Framing Error) : positionné à 1 par hardware lorsque se produit une erreur liée à une surcharge ou une erreur de format. Une surcharge peut se définir par le fait que plusieurs caractères ont été reçus avant la lecture du caractère précédent. Une erreur de format correspond à une perte de synchronisation avec le compteur de bits (par exemple absence du premier bit STOP). Il est à noter que cet indicateur ORFE ne concerne que le registre récepteur. Cet indicateur correspondant à deux types d'erreurs différents (c'est-à-dire quatre états), il est donc nécessaire d'utiliser un second bit pour distinguer ces deux erreurs : on utilisera donc conjointement avec ORFE, le bit 7 RDRF.
Si RDRF = ORFE = 1 alors, c'est une erreur de surcharge qui s'est produite.
Si RDRF = 0 et ORFE = 1 alors, il s'agit d'une erreur de format.
Tant que RDRF restera positionné à 1, aucun nouveau caractère ne pourra être transféré dans le registre récepteur. ORFE est remis à 0 par une nouvelle lecture du registre d'état et du registre récepteur ou par une réinitialisation ($\overline{\text{RESET}}$).

• ***bit 7 - RDRF*** (Receive Data Register Full) : positionné à 1 par hardware lorsque le contenu du registre à décalage de réception est assemblé et que le transfert vers le registre récepteur est opéré. RDRF est remis à 0 par une nouvelle lecture du registre d'état et du registre récepteur ou par une réinitialisation ($\overline{\text{RESET}}$).

Nous pouvons ainsi résumer la signification des bits du registre d'état.

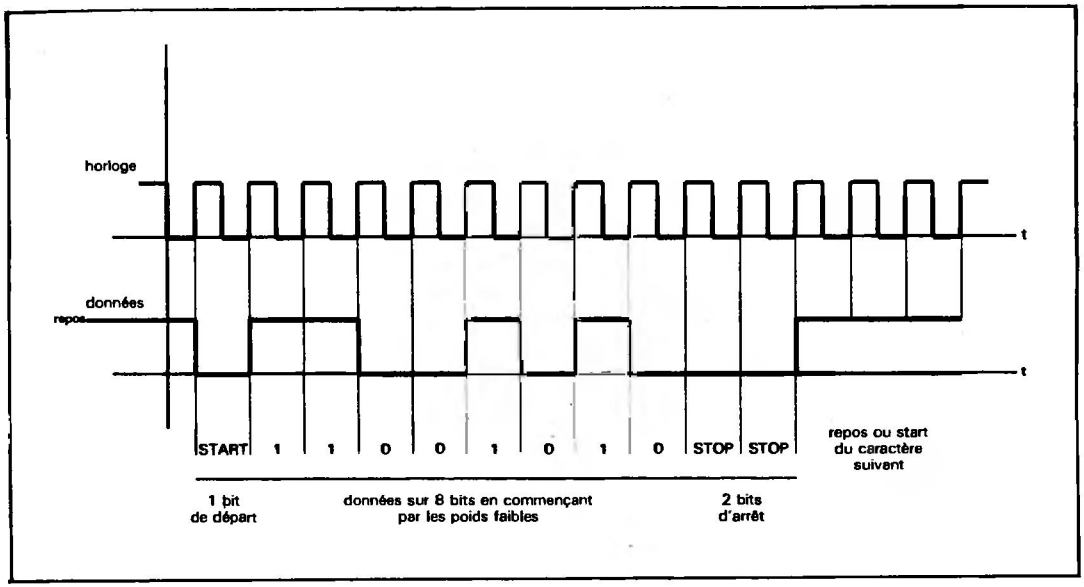
indicateur	bit	signification
WU	0	sélection du processus d'activation
TE	1	transmission autorisée
TIE	2	interruption autorisée si registre transmetteur vide
RE	3	réception autorisée
RIE	4	interruption demandée si erreur en réception ou registre récepteur plein
TDRE	5	registre transmission vide (caractère envoyé)
ORFE	6	erreur de format (sur bit START ou STOP) ou débordement
RDRF	7	registre réception plein (caractère assemblé)

PROCÉDURE DE TRANSMISSION START/STOP

La procédure de transmission employée sur ALICE est la procédure START/STOP.
Le format des données transmises est le suivant :

- 1 bit de départ (START)
- 8 bits de données
- 2 bits d'arrêt (STOP)

Ce qui donne le schéma suivant pour la transmission du caractère "S" dont le code ASCII est 83 en décimal, soit 53 en hexadécimal ou 01010011 en binaire.



LE REGISTRE DE COMMANDE

Ce registre contrôle le fonctionnement du transmetteur et du récepteur. Il s'agit d'un registre à écriture seule dans lequel les paramètres suivant pourront être modifiés.

- vitesse de transmission (ou de réception)
- format
- horloge utilisée (interne ou externe)
- bit 2 du port 2

En fait, seulement 4 bits de ce registre sont utilisés. Ces quatre bits sont initialisés à 0 par un RESET. On peut considérer qu'il s'agit de deux paires de bits. En effet, les bits 0 et 1 sont utilisés pour contrôler, pour la source horloge interne, la vitesse ; alors que les bits 2 et 3 permettent de sélectionner l'horloge utilisée (interne ou externe) ainsi que le format.
On obtient le schéma suivant :

X	X	X	X	CC_1	CC_0	S_1	S_0
b7	b6	b5	b4	b3	b2	b1	b0

registre de commande (adresse '10)

• ***bits O et 1*** (Speed Select) : ces deux bits permettent de sélectionner la vitesse pour l'horloge interne. Les quatre débits qui peuvent être sélectionnés sont fonction de la fréquence d'horloge 02 du MPU.

		XTAL	4,0 MHz	4,9152 MHz	2,4576 MHz
S1	*S2*	*02*	*1,0 µs/1,0 MHz*	*814 ms/1,2288 MHz*	*1,63 µs/614,4 kHz*
0	0	÷16	16 µs/62 500 bauds	13 µs/76 800 bauds	26 µs/38 400 bauds
0	1	÷128	128 µs/7 812,5 bauds	104,2 µs/9 600 bauds	20B µs/4 800 bauds
1	0	÷1024	1,024 ms/976,6 bauds	833,3 µs/1 200 bauds	1,67 ms/600 bauds
1	1	÷4096	4,096 ms/244,1 bauds	3,33 ms/300 bauds	6,67 ms/150 bauds

• ***bit 2 et 3*** (Clock Control and Format Select) : ces deux bits contrôlent le format (start/stop ou bi-phase) et sélectionnent la source horloge (interne ou externe). Si on choisit l'horloge externe, les bits S1 et So sont inopérants.

CC1	CCo	format	horloge	port2/bit2
0	0	bi-phase	interne	———
0	1	Start/Stop	interne	———
1	0	Start/Stop	interne	sortie
1	1	Start/Stop	externe	entrée

LE REGISTRE DE TRANSMISSION

Ce registre reçoit, directement du bus de données, le caractère qui doit être transmis. Ce caractère est alors envoyé dans le registre à décalage de transmission pour "sérialiser" la donnée.

LE REGISTRE DE RÉCEPTION

Ce registre reçoit le caractère à lire par le microprocesseur après qu'il ait été assemblé par le registe à décalage de réception.

LES OPÉRATIONS SÉRIE

Avant de faire une entrée/sortie série, il est nécessaire que l'interface soit initialisée, c'est-à-dire :
- écrire dans le registre de commande, le mot de commande mettant à jour les bits spécifiant le débit et le format à respecter pour l'échange.
- écrire les bits désirés dans le registre d'état. Les bits TE et RE doivent être positionnés à 1.

OPÉRATION DE TRANSMISSION

L'opération de transmission est autorisée par le bit TE du registre d'état. Lorsque ce bit est positionné à 1, le contenu du registre à décalage de transmission, dans lequel est sérialisée la donnée, est envoyé sur le bit 4 du port 2 ; lorsque TE est à 1, le bit 4 du registre de direction du port 2 est automatiquement positionné à 1, de façon à permettre la sortie. La mise à 1 du bit TE initialise la sortie série avec l'envoi de 10 bits consécutifs à 1 qui permettent la synchronisation sur l'horloge. Ensuite, les différents caractères qui seront transmis le seront en asynchrone.
Deux cas peuvent alors se présenter :

- si le bit TDRE (registre de transmission vide) est à 1, une chaine continue de bits à 1 est envoyée, indiquant que la ligne est au repos.
- si une donnée a été chargée dans le registre de transmission (qui est alors plein), TDRE est alors mis à 0 et le contenu du registre de transmission, qui joue ici le simple rôle d'un tampon, est envoyé sur son registre à décalage de façon à sérialiser la donnée pour pouvoir l'envoyer vers un périphérique par exemple.

Notons que toute transmission série d'une donnée est composée d'abord d'un bit de départ (valeur "0") transmis en premier, puis des huits bits du caractère constituant la donnée, en commençant par les poids faibles, puis enfin de deux bits d'arrêt (valeur "1"). Le bit TDRE est positionné à 1 au moment du transfert du contenu du registre de transmission vers son registre à décalage associé. Cette mise à jour de TDRE est réalisée directement par le hard.
Si le programme ne répond pas dans les délais (TDRE est alors toujours à 1 lorsque le transfert du caractère suivant vers le registre de transmission se produit), alors un bit à "1" est envoyé en place de "0" comme bit de départ. Aucun 0 ne peut être envoyé tant que TDRE est positionné à 1. Il est à noter également que l'interface série du 6803 ne fait pas apparaître de bit de parité.

OPÉRATION DE RÉCEPTION

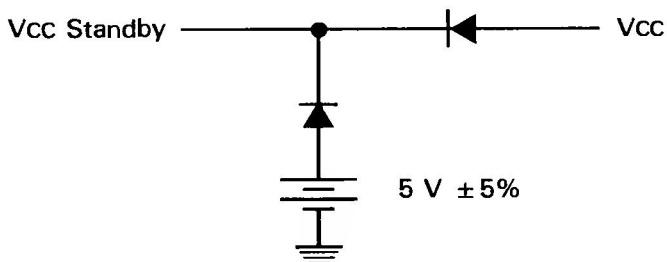
La réception est autorisée lorsque le bit RE (Receive Enable), du registre d'état est positionné à 1. La donnée est alors entrée par l'intermédiaire du bit 3 du port 2. Les opérations de réception sont supervisées par le registre d'état et par le registre de commande.
L'intervalle correspondant à un bit de donnée est divisé en huit sous-intervalles, permettant ainsi la synchronisation interne. La chaîne de bits reçue est synchronisée sur l'horloge de réception par le premier bit à 0 (bit de départ) suivant un signal de ligne au repos (suite continue de bits à 1 sur la ligne).
Au milieu approximatif du temps correspondant à la tramsmission de chaque bit, un échantillonnage a lieu, pendant les dix bits suivants un bit de départ.
Si le dixième bit n'est pas à 1 (bit d'arrêt), alors il y a une erreur de format et le drapeau ORFE est positionné à 1 dans le registre d'état. Par contre, si le dixième bit est à 1, la donnée est transmise au registre de réception dans lequel le MPU viendra la lire, et l'indicateur RDRF est positionné à 1 dans le registre d'état. Si RDRF est toujours à 1 après un délai correspondant à la transmission de dix bits, alors ORFE sera positionné à 1, indiquant qu'un débordement se produit sur le registre de réception : il s'agit du cas où le MPU n'a pas prélevé à temps le contenu du registre de réception. Cette lecture par le MPU, précédée d'une lecture du registre d'état a pour effet de réinitialiser à 0 les indicateurs RDRF et ORFE.
Comme nous venons de le voir, le rôle des registres de transmission et de réception est en fait un rôle de tampon entre le MPU et les registres à décalage correspondants.

LE REGISTRE DE CONTRÔLE DE LA RAM

Ce registre, situé à l'adresse '014, a une utilisation bien particulière. Il a en effet pour fonction essentielle de donner des informations d'état sur les opérations de conservation de la RAM. En fait, seuls les deux bits de poids fort sont utilisés dans ce registre.

STB PWR	RAME	X	X	X	X	X	X	adresse '014
b7	b6	b5	b4	b3	b2	b1	b0	

A ce moment il est nécessaire de revenir sur la signification de la broche Vcc Standby du 6803. On a vu que cette broche pouvait être utilisée pour fournir au 6803, en cas de coupure de l'alimentation, une alimentation auxiliaire de 5 V (± 5%) qui permet de conserver le contenu des 64 premiers octets de la RAM, internes au 6803 avec un courant de 8 mA maximum. On peut utiliser le schéma suivant pour réaliser la connection de la batterie sur le 6803.

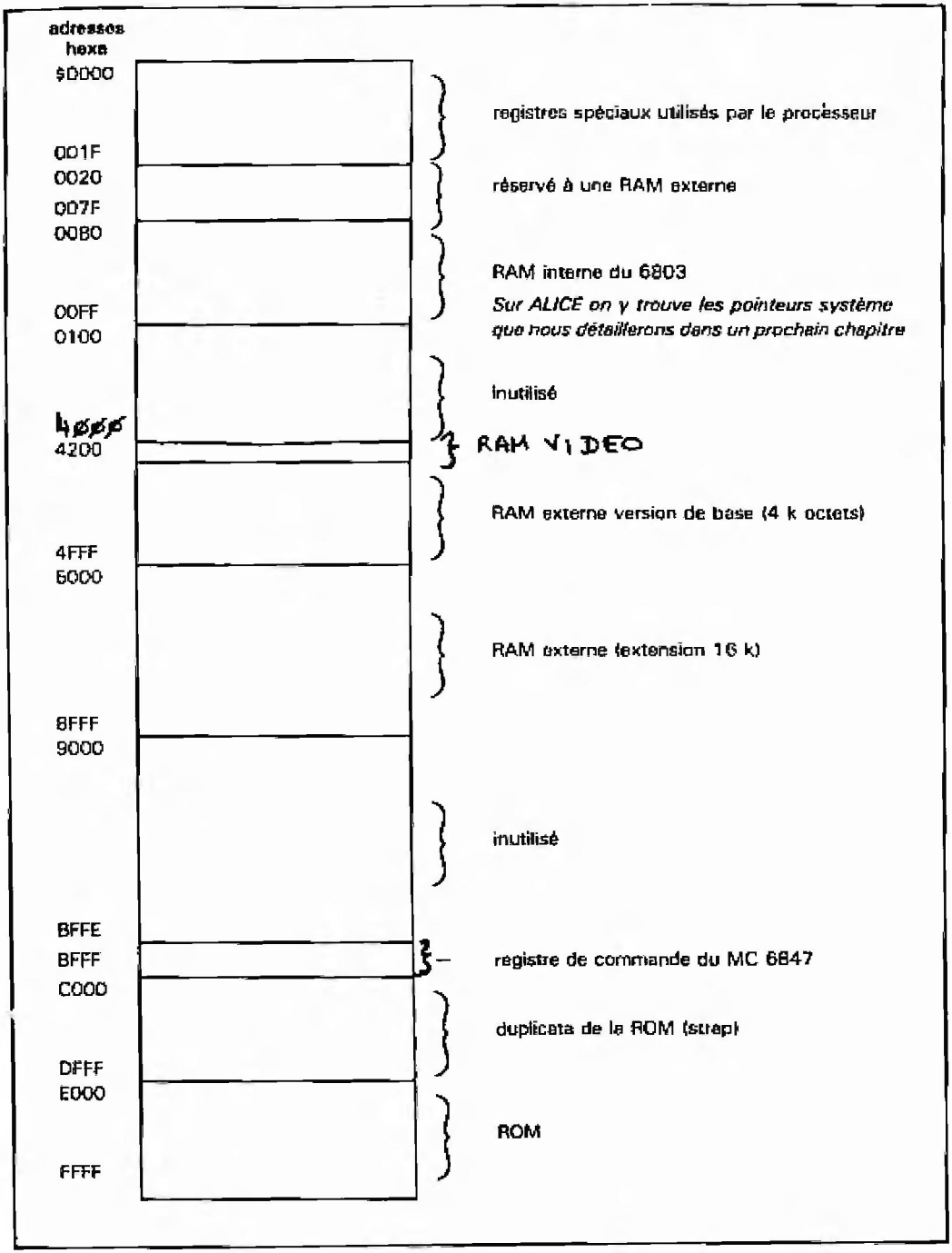


Le bit 6 et le bit 7 de l'adresse '014, c'est-à-dire du registre de contrôle RAM vont alors être utilisés :

• **bit 6 - RAME** (Ram Enable) : permet à l'utilisateur de rendre impossible l'accès à la RAM interne. Ce bit est positionné à "1" lors de l'initialisation ou par $\overline{\text{RESET}}$, ce qui permet l'utilisation de la RAM interne, mais il peut être mis à "0" ou à "1" par le programme. Lorsque la RAM interne est déconnectée, les données sont prélevées dans une mémoire externe.

• **bit 7 - STB PWR** (Standby Power) : ce bit est positionné à "0" à chaque fois que la tension de secours diminue au dessous du minimum. Ce bit est un indicateur qui peut être utilisé pour détecter une baisse de tension en dessous du niveau minimum de Vcc Standby. Ce bit peut être mis à 1 par le programme mais il n'est pas affecté par $\overline{\text{RESET}}$.

Nous savons donc que le 6803 peut adresser jusqu'à 64 k octets de mémoire. Or 128 octets sont internes au 6803, en plus des 32 premiers octets qui sont utilisés comme registres spéciaux, et que nous venons de voir. On obtient alors un mapping de la mémoire qui est le suivant, sur ALICE :


adresses hexa
$0000
001F
0020
007F
0080
00FF
0100
4000
4200
4FFF
6000
8FFF
9000
BFFE
BFFF
C000
DFFF
E000
FFFF
registres spéciaux utilisés par le processeur
réservé à une RAM externe
RAM interne du 6803
Sur ALICE on y trouve les pointeurs système
que nous détaillerons dans un prochain chapitre
inutilisé
RAM VIDEO
RAM externe version de base (4 k octets)
RAM externe (extension 16 k)
inutilisé
registre de commande du MC 6847
duplicata de la ROM (strap)
ROM

Les adresses '020 à 07F sont réservées à une RAM externe à laquelle il est possible d'accéder grâce au bit RAME du registre situé à l'adresse '014. Les 64 premiers octets, internes au 6803, seront alors sauvegardés grâce à une batterie de sauvegarde, pendant que l'on pourra utiliser une RAM externe de 128 octets maximum (comme par exemple une MC 6810 A de MOTOROLA), de façon à permettre la sauvegarde des informations essentielles contenues dans les registres spéciaux, nécessaires au bon fonctionnement de l'ordinateur. Il est à noter que cette RAM externe n'est pas disponible sur ALICE.

RÉSUMÉ DES REGISTRES SPÉCIAUX

adresse hexa	registre
00	registre de direction associé au port 1 d'E/S
01	registre de direction associé au port 2 d'E/S
02	port 1 d'E/S
03	port 2 d'E/S
08	registre d'état du temporisateur
09	octet de poids fort du compteur
0A	octet de poids faible du compteur
0B	octet de poids fort sortie temporisateur
0C	octet de poids faible sortie temporisateur
0D	octet de poids fort entrée temporisateur
0E	octet de poids faible entrée temporisateur
10	registre de commande format et débit interf. série
11	registre d'état de l'interface série
12	registre de réception série
13	registre de transmission série
14	registre de contrôle de la RAM
15 à 1F	réservé

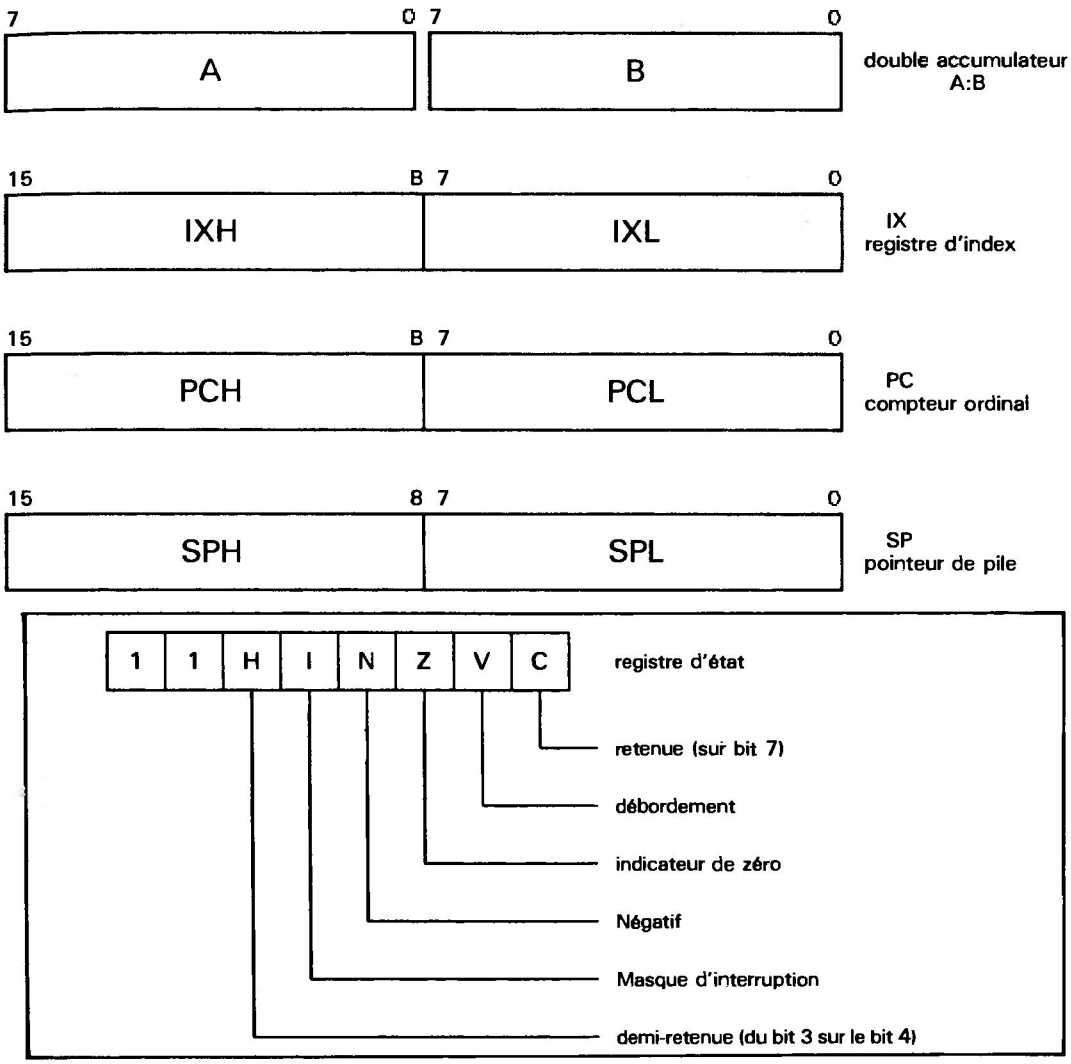
LE LANGAGE MACHINE DU 6803

Le MC 6803 possède un jeu de 83 instructions, avec 7 modes d'adressage. Il intègre toutes les instructions du 6800 ce qui lui permet de recevoir, et d'exécuter sans transformation des programmes écrits en assembleur 6800. Par rapport à ce dernier, on peut remarquer une relative diminution du nombre de cycles pour un certain nombre d'instructions de façon à réduire les temps d'exécution. Ainsi les instructions en adressage indexé perdent-elles toutes un cycle par rapport au 6800. De plus, le 6803 intègre un certain nombre d'instructions nouvelles :

ABX - ADDD - ASLD - LDAD - LSRD - MUL - PSHX - STAD - SUBD

qui sont toutes des instructions travaillant sur seize bits, en considérant comme accumulateur de 16 bits le double accumulateur A:B, c'est-à-dire la concaténation de A et de B. De plus, on note une instruction très intéressante : MUL qui est une multiplication hardware d'une donnée de 8 bits par une seconde donnée de huit bits, plaçant le résultat sur 16 bits dans le double registre A:B. Notons à cet effet que le double accumulateur est bien le résultat de la concaténation de l'accumulateur A et de l'accumulateur B et que, par voie de conséquence, toute utilisation du double accumulateur a pour effet de détruire les contenus des accumulateurs A et B.

On peut donc schématiser les registres du MC 6803 de la façon suivante :



LES MODES D'ADRESSAGE DU 6803

Nous avons vu que le MC 6803 offrait 7 modes d'adressage. Nous allons donc rapidement les décrire. Il s'agit de :

- adressage accumulateur
- adressage immédiat
- adressage direct
- adressage étendu
- adressage indexé
- adressage implicite
- adressage relatif

• ***Adressage accumulateur*** : il s'agit d'adressage portant soit sur l'accumulateur A, soit sur l'accumulateur B. Ces instructions sont codées sur un seul octet, l'opérande étant ici l'accumulateur (ex. : ROLA : rotation à gauche de l'accumulateur A).

• ***Adressage immédiat*** : le signe indiquant ce mode d'adressage en language source est # après le code opération. Dans ce mode d'adressage, l'opérande est contenue dans le deuxième octet de l'instruction, à l'exeption de LDS - LDX - CPX - SBD et LDAD, pour lesquelles l'opérande est codé sur deux octets et est donc contenu dans le deuxième et le troisième octet de l'instruction. Les instructions en mode immédiat sont donc codées sur deux ou trois octets.

• ***Adressage direct*** : les instructions traduites en langage objet comportent deux octets en adressage direct. Le deuxième octet de l'instruction contient alors l'adresse effective à laquelle se trouve la donnée. Ce mode d'adressage permet donc au programme d'adresser directement les 256 premiers octets ('00 à 'FF), de la mémoire morte (RAM interne du 6803 sur ALICE).

• ***Adressage étendu*** : comme en adressage direct, l'instruction contient l'adresse effective à laquelle se trouve la donnée. En adressage direct (souvent appelé sur d'autres machines adressage page zéro), on ne peut adresser, sur 8 bits, que les 256 premiers octets de la mémoire. En adressage étendu (encore appelé adressage absolu), l'adresse sera codée sur deux octets, ce qui permettra d'adresser les 2^{16} = 64 k RAM possibles sur 6803. Dans ce type d'adressage, l'adresse de la donnée est codée dans les deuxième et troisième octets de l'instruction ; le deuxième octet du code objet est donc le poids fort de l'adresse et le troisième octet est le poids faible de l'adresse. En mode étendu, les instructions sont toujours codées sur trois octets en code objet.

• ***Adressage indexé*** : en mode indexé, le contenu du deuxième octet de l'instruction constitue un déplacement absolu qui est ajouté au contenu du registre IX pour obtenir l'adresse effective de la donnée. Ce déplacement sur huit bits est ajouté à l'octet de poids faible de IX puis la retenue est rajoutée à l'octet de poids fort de IX ; le résultat est conservé dans une zone temporaire de telle sorte que le contenu du registre d'index n'est pas altéré. L'assembleur sélectionne ce mode d'adressage toutes les fois que le champ opérande de l'instruction se présente sous la forme suivante (valeur sur huit bits , X). Ce mode d'adressage permettra très facilement d'accéder séquentiellement à une suite de valeurs rangées dans une table. En mode indexé, les instructions sont codées sur deux octets en code objet.

• ***Adressage implicite*** : dans ce mode d'adressage, l'adresse de l'opérande est implicitement contenue dans l'instruction (par exemple le pointeur de pile, le registre d'index). Il s'agit d'instructions qui seront codées sur un seul octet. On peut d'ailleurs considérer l'adressage accumulateur comme une forme d'adressage implicite. Notons qu'en adressage implicite, le code opération contient également implicitement l'adresse à laquelle sera stocké le résultat de l'opération (par exemple ASLD : décalage arithmétique à gauche de A:B, résultat dans A:B).

• ***Adressage relatif*** : en adressage relatif, le contenu du deuxième octet de l'instruction est considéré comme un entier signé qui est ajouté au contenu du compteur ordinal après l'avoir incrémenté de deux. Ce mode d'adressage est donc réservé aux branchements. L'adresse finale du branchement est donc définie comme la somme de l'adresse de l'instruction suivant le branchement et du contenu du deuxième octet de l'instruction de branchement. Ce deuxième octet étant considéré comme un entier signé, c'est-à-dire positif s'il est inférieur à 129 et négatif s'il est supérieur ou égal à 129, ce qui permet d'effectuer des branchements en avant ou en arrière en respectant ces limites. Ces instructions nécessitent donc deux octets en code objet.

Pour terminer ce chapitre consacré au microprocesseur 6803 qui est le "cœur " d'ALICE, nous allons maintenant étudier le jeu complet d'instructions de ce processeur en précisant pour chaque instruction, son mode de fonctionnement, les indicateurs qui sont affectés par son exécution ainsi que l'expression booléenne qui permet de déterminer la nouvelle valeur de ces indicateurs. Enfin, pour chaque instruction, un tableau présentera les mnémoniques employés en assembleur, les modes d'adressage, le nombre d'octets nécessaires en code objet ainsi que les valeurs décimale et hexadécimale du code opération. Par ailleurs, on trouvera dans ce tableau le nombre de cycles d'horloge nécessaires à l'exécution de chaque instruction. Compte-tenu de la fréquence d'horloge utilisée sur ALICE, on pourra assimiler le cycle à 1 ms.
Cette partie sera sans doute très utile à tous ceux, et je pense qu'ils sont nombreux, qui souhaitent écrire des programmes en langage machine sur ALICE ou MC10.

LES INSTRUCTIONS DU MC 6803

Abréviations utilisées dans les pages suivantes.
L'auteur a essayé d'employer au maximum les abréviations standard dont la liste est la suivante :

abréviations	définitions
( )	contenu de
A, B	accumulateurs
:	concaténation (par exemple A:B)
M	adresse mémoire spécifiée dans le programme
M+1	adresse suivante
CC	registre d'état
SP	pointeur de pile
IX	registre d'index
IXH	octet de poids fort du registre d'index
IXL	octet de poids faible du registre d'index
PC	compteur ordinal
PCH	octet de poids fort du compteur ordinal
PCL	octet de poids faible du compteur ordinal
SP	pointeur de pile
SPH	octet de poids fort du pointeur de pile
SPL	octet de poids faible du pointeur de pile
'	hexadécimal
R	résultat d'une opération
Ri	bit "i" du résultat
↓	empiler
↑	dépiler
←	assignation
+	addition
×	multiplication
÷	division
−	soustraction
V ou +	OU logique
⊕	OU exclusif
•	ET logique

ABA : somme de l'accumulateur A et de l'accumulateur B.

Opération :

$A \leftarrow [A] + [B]$

Description :

effectue la somme des contenus des accumulateurs A et B, et place le résulat dans A.

Indicateurs affectés :

H : mis à 1 s'il y a une retenue sur le bit 3, sinon remis à 0
N : mis à 1 si le bit 7 du résultat est à 1, sinon remis à 0
V : mis à 1 s'il y a débordement en complément à 2 du résultat, sinon remis à 0
C : mis à 1 s'il y a une retenue du bit 7 du résultat, sinon remis à 0

Expression booléenne :

$H = A_3 \cdot B_3 \vee B_3 \cdot \overline{R}_3 \vee \overline{R}_3 \cdot A_3$
$N = R_7$
$Z = \overline{R}_7 \cdot \overline{R}_6 \cdot \overline{R}_5 \cdot \overline{R}_4 \cdot \overline{R}_3 \cdot \overline{R}_2 \cdot \overline{R}_1 \cdot \overline{R}_0$
$V = A_7 \cdot B_7 \cdot \overline{R}_7 \vee \overline{A}_7 \cdot \overline{B}_7 \cdot R_7$
$C = A_7 \cdot B_7 \vee B_7 \cdot \overline{R}_7 \vee \overline{R}_7 \cdot A_7$

Codes :

modes d'adressage	nombre de cycles	nombre d'octets	codes machine	
			hexa	décimal
implicite	2	1	1B	027

ABX : ajoute l'accumulateur B au registre IX.

Opération :

IX ← [IX] + [B]

Description :

ajoute au contenu du registre d'index IX, le contenu de l'accumulateur B en tenant compte de l'éventuelle retenue provenant de l'octet de poids faible de IX, et place le résultat dans IX.

Indicateurs affectés :

aucun.

Codes :

modes d'adressage	nombre de cycles	nombre d'octets	codes machine	
			hexa	décimal
implicite	3	1	3A	058

ADC : somme avec retenue.

Opération :

$Acc \leftarrow (Acc) + (M) + (C)$

Description :

ajoute la valeur du bit C à la somme des contenus d'un accumulateur et de M et place le résultat dans cet accumulateur.

Indicateurs affectés :

H : mis à 1 s'il y a une retenue du bit 3 sur le bit 4, sinon remis à 0
N : mis à 1 si le bit 7 du résultat vaut 1, sinon remis à 0
V : mis à 1 s'il y a débordement en complément à 2 sur le résultat, sinon remis à 0
C : mis à 1 s'il y a une retenue du bit 7 du résultat, sinon remis à 0

Expression booléenne :

$H = Acc_3 \cdot M_3 + M_3 \cdot \overline{R_3} + \overline{R_3} \cdot Acc_3$
$N = R_7$
$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$
$V = Acc_7 \cdot M_7 \cdot \overline{R_7} + \overline{Acc_7} \cdot \overline{M_7} \cdot R_7$
$C = Acc_7 \cdot M_7 + M_7 \cdot \overline{R_7} + \overline{R_7} \cdot Acc_7$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
ADC A	immédiat	2	2	89	137
ADC A	direct	3	2	99	153
ADC A	étendu	4	3	B9	185
ADC A	indexé	4	2	A9	169
ADC B	immédiat	2	2	C9	201
ADC B	direct	3	2	D9	217
ADC B	étendu	4	3	F9	249
ADC B	indexé	4	2	E9	233

ADD : addition sans retenue.

Opération :

ACC ← (ACC) + (M)

Description :

ajoute le contenu de M au contenu de l'un des accumulateurs et place le résultat dans cet accumulateur.

Indicateurs affectés :

H : mis à 1 s'il y a une retenue du bit 3 sur le bit 4, sinon remis à 0
N : mis à 1 si le bit 7 du résultat vaut 1, sinon remis à 0
Z : mis à 1 si le résultat est nul, sinon remis à zéro
V : mis à 1 s'il y a débordement en complément à 2 sur le résultat, sinon remis à 0
C : mis à 1 s'il y a une retenue du bit 7 du résultat, sinon remis à 0

Expression booléenne :

$H = ACC_3 \cdot M_3 + M_3 \cdot \overline{R_3} + \overline{R_3} \cdot ACC_3$
$N = R_7$
$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$
$V = ACC_7 \cdot M_7 \cdot \overline{R_7} + \overline{ACC_7} \cdot \overline{M_7} \cdot R_7$
$C = ACC_7 \cdot M_7 + M_7 \cdot \overline{R_7} + \overline{R_7} \cdot ACC_7$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
ADD A	immédiat	2	2	8B	139
ADD A	direct	3	2	9B	155
ADD A	étendu	4	3	BB	187
ADD A	indexé	4	2	AB	171
ADD B	immédiat	2	2	CB	203
ADD B	direct	3	2	DB	219
ADD B	étendu	4	3	FB	251
ADD B	indexé	4	2	EB	235

ADDD : somme sur deux octets.

Opération :

A:B ← (A:B) + (M:M+1)

Description :

ajoute au contenu du double accumulateur A:B (sur 16 bits), le contenu de M:M+1 et place le résultat dans A:B.

Indicateurs affectés :

N : mis à 1 si le bit 15 du résultat vaut 1, sinon remis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à zéro.
V : mis à 1 s'il y a débordement en complément à 2 sur le résultat, sinon remis à 0
C : mis à 1 s'il y a une retenue sur le bit le plus significatif du résultat, sinon remis à 0

Expression booléenne :

$N = R_{15}$
$Z = \overline{R}_{15} \cdot \overline{R}_{14} \cdot \overline{R}_{13} \cdot \overline{R}_{12} \cdot \overline{R}_{11} \cdot \overline{R}_{10} \cdot \overline{R}_{9} \cdot \overline{R}_{8} \cdot \overline{R}_{7} \cdot \overline{R}_{6} \cdot \overline{R}_{5} \cdot \overline{R}_{4} \cdot \overline{R}_{3} \cdot \overline{R}_{2} \cdot \overline{R}_{1} \cdot \overline{R}_{0}$
$V = A_7 \cdot M_7 \cdot \overline{R}_{15} + \overline{A}_7 \cdot \overline{M}_7 \cdot R_{15}$
$C = A_7 \cdot M_7 + M_7 \cdot \overline{R}_{15} + \overline{R}_{15} \cdot \overline{A}_7$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
ADDD	immédiat	4	3	C3	195
ADDD	direct	5	2	D3	211
ADDD	étendu	6	3	F3	243
ADDD	indexé	6	2	E3	227

AND : ET logique.

Opération :

$ACC \leftarrow (Acc) \cdot (M)$

Description :

l'opération logique "ET" est réalisée entre le contenu de M et le contenu de l'un des accumulateurs. Le résultat est rangé dans ce même accumulateur.

Indicateurs affectés :

N : mis à 1 si le bit 7 du résultat vaut 1, sinon remis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à zéro
V : mis à 0

Expression booléenne :

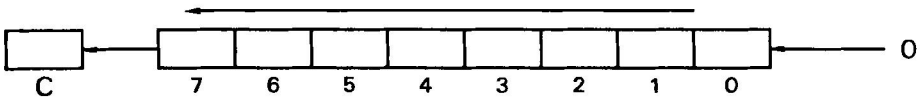
$N = R_7$
$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$
$V = 0$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
AND A	immédiat	2	2	84	132
AND A	direct	3	2	94	148
AND A	étendu	4	3	B4	180
AND A	indexé	4	2	A4	164
AND B	immédiat	2	2	C4	196
AND B	direct	3	2	D4	212
AND B	étendu	4	3	F4	244
AND B	indexé	4	2	E4	228

ASL : décalage arithmétique à gauche.
ou LSL

Opération :



Description :

décale d'un bit vers la gauche le contenu d'un accumulateur ou d'un mot mémoire. Le bit 0 est mis à 0 et le bit C reçoit le contenu du bit 7.

Indicateurs affectés :

N : mis à 1 si le bit 7 du résultat vaut 1, sinon remis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à 0
V : mis à 1 si, après décalage, soit (N est à 1 et C est à 0), soit (N est à 0 et C est à 1) ; sinon remis à 0
C : mis à 1 si, avant le décalage, le bit 7 était à 1, sinon mis à 0

Expression booléenne :

$N = R_7$
$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$
$V = N \oplus C$ (OU exclusif de N et C)
$C = M_7$ (ou ACC_7)

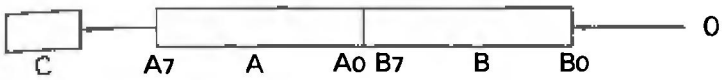
Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
ASL A	implicite	2	1	48	072
ASL B	implicite	2	1	58	088
ASL	étendu	6	3	78	120
ASL	indexé	6	2	68	104

ASLD : décalage à gauche sur double accumulateur.

Opération :

A:B ← 2 × (A:B)



Description :

décalage à gauche du contenu sur 16 bits du double accumulateur A:B. Le bit 0 de B est mis à 0 tandis que le bit C reçoit le contenu du bit 7 de A.

Indicateurs affectés :

N : mis à 1 si le bit 15 du résultat vaut 1, sinon remis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à 0
V : mis à 1 si, après décalage, soit (N est à 1 et C à 0), soit (N est à 0 et C à 1), sinon remis à 0
C : mis à 1 si, avant décalage, le bit 7 de A était à 1, sinon remis à 0

Expression booléenne :

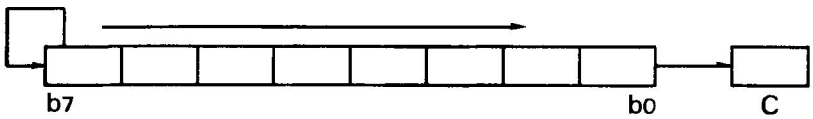
$N = R_{15}$
$Z = \overline{R}_{15} \cdot \overline{R}_{14} \cdot \overline{R}_{13} \ldots \overline{R}_{2} \cdot \overline{R}_{1} \cdot \overline{R}_{0}$
$V = N \oplus C$ (OU exclusif de N et C)
$C = A_7$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
ASLD	implicite	3	1	05	05

ASR : décalage arithmétique à droite.

Opération :



Description :

décale d'un bit vers la droite le contenu de l'un des accumulateurs ou d'un mot mémoire. Le bit 7 conserve sa valeur tandis que C reçoit le contenu du bit 0.

Indicateurs affectés :

N : mis à 1 si le bit 7 du résultat est à 1, sinon remis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à 0
V : mis à 1 si, après décalage, soit (N est à 1 et C à 0), soit (N est à 0 et C à 1), sinon remis à 0
C : mis à 1 si, avant décalage, le bit 0 était à 1, sinon remis à 0

Expression booléenne :

$N = R_7$
$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$
$V = N \oplus C$ (OU exclusif de N et C)
$C = M_0$ (ou ACC0)

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
LSR A	implicite	2	1	47	071
LSR B	implicite	2	1	57	087
LSR	étendu	6	3	77	119
LSR	indexé	6	2	67	103

BCC : branchement si bit C à 0.
ou BHS

Opération :

PC ← (PC) + 02 + déplacement si (C) = 0

Description :

teste la valeur de C et effectue un branchement si C est à 0.

Indicateurs affectés

aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
BHS ou BCC	relatif	3	2	24	036

BCS : branchement si bit C à 1.
ou BLO

Opération :
PC ← (PC) + 02 + déplacement si (C) = 1

Description :
teste la valeur de C et effectue un branchement si C est à 1.

Indicateurs affectés :
aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
BLO ou BCS	relatif	3	2	25	037

BEQ : branchement si égalité.

Opération :
PC ← (PC) + 02 + **déplacement si (Z) = 1**

Description :
teste la valeur de Z et effectue un branchement si Z est à 1.

Indicateurs affectés :
aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
BEQ	relatif	3	2	27	039

BGE : branchement si supérieur ou égal à 0.

Opération :
PC ← (PC) + 02 + déplacement si (N) ⊕ (V) = 0

Description :
branchement si : soit (N est à 1 et V est à 1), soit (N est à 0 et V est à 0).

Indicateurs affectés :
aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
BGE	relatif	3	2	2C	044

BGT : branchement si supérieur à 0.

Opération :
PC ← (PC) + 02 + déplacement si (Z) • [(N) ⊕ (V)] = 0

Description :
branchement si [Z est à 0] **et** [soit (N est à 1 et V est à 1), soit (N est à 0 et V est à 0)].

Indicateurs affectés :
aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
BGT	relatif	3	2	2E	046

BHI : branchement si supérieur.

Opération :
PC ← (PC) + 02 + déplacement si (C) + (Z) = 0

Description :
branchement si C et Z sont tous les deux à zéro.

Indicateurs affectés :
aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
BHI	relatif	3	2	22	034

BIT : ET logique, bit par bit.

Opération

(ACC) • (M)

Description :

réalise le "ET" logique entre le contenu de l'un des accumulateurs et le contenu d'un mot mémoire, et modifie les indicateurs en conséquence. Ni le contenu de l'accumulateur, ni le contenu de M ne sont modifiés.

Indicateurs affectés :

N : mis à 1 si le bit le plus significatif du résultat du "ET" doit être 1, sinon remis à 0
Z : mis à 1 si le résultat du "ET" doit être 0, sinon remis à 0.
V : mis à 0

Expression booléenne :

$N = R7$
$Z = \overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$
$V = 0$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
BIT A	immédiat	2	2	85	133
BIT A	direct	3	2	95	149
BIT A	étendu	4	3	B5	181
BIT A	indexé	4	2	A5	165
BIT B	immédiat	2	2	C5	197
BIT B	direct	3	2	D5	213
BIT B	étendu	4	3	F5	245
BIT B	indexé	4	2	E5	229

BLE : branchement si inférieur ou égal à zéro.

Opération :
PC ← (PC) + 02 + déplacement si (Z) + [(N) ⊕ (V)] = 1

Description :
branchement si (Z est à 1) **ou** [soit (N est à 1 et V à 0) ou soit (N est à 0 et V est à 1)].

Indicateurs affectés :
aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
BLE	relatif	3	2	2F	047

BLS : branchement si inférieur ou égal.

Opération :
PC ← (PC) + 02 + déplacement si (C) + (Z) = 1

Description :
branchement si C est à 1 ou Z est à 1.

Indicateurs affectés :
aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
BLS	relatif	3	2	23	035

BLT : branchement si inférieur à zéro.

Opération :
PC ← (PC) + 02 + déplacement si (N) ⊕ (V) = 1

Description :
branchement si (N est à 1 et V est à 0) ou (N est à 0 et V à 1).

Indicateurs affectés :
aucun.

Codes

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
BLT	relatif	3	2	2D	045

BMI : branchement si moins.

Opération :
PC ← (PC) + 02 + déplacement si (N) = 1 (résultat < 0)

Description :
branchement si N est à 1.

Indicateurs affectés :
aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
BMI	relatif	3	2	2B	043

BNE : branchement si différent de 0.

Opération :
PC ← (PC) + 02 + déplacement si (Z) = 0

Description :
branchement si Z est à 0.

Indicateurs affectés :
aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
BNE	relatif	3	2	26	038

BPL : branchement si plus.

Opération :
PC ← (PC) + 02 + déplacement si (N) = 0

Description :
branchement si N est à 0.

Indicateurs affectés :
aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
BPL	relatif	3	2	2A	042

BRA : branchement inconditionnel.

Opération :
PC ← (PC) + 02 + déplacement

Description :
branchement inconditionnel à l'adresse calculée par la formule ci-dessus. Le déplacement est un nombre signé (entre -128 et +127).

Indicateurs affectés :
aucun.

BRN : Ne pas brancher. ⇔ NOP

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
BRA	relatif	3	2	20	032

BSR : branchement à un sous-programme.

Opération :

PC ← (PC) + 02
empile OMS de (PC)
SP ← (SP) − 01
empile OPS de (PC)
SP ← (SP) − 01
PC ← (PC) + déplacement

Description :

le compteur ordinal est incrémenté de 2. L'octet le moins significatif du contenu du compteur ordinal est emplié et le pointeur de pile est décrémenté de 1. Puis l'octet le plus significatif du contenu du compteur ordinal est empilé et le pointeur de pile est de nouveau décrémenté de 1. Enfin, un branchement est effectué à l'adresse calculée en ajoutant le déplacement au contenu de PC.

Indicateurs affectés :

aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
BSR	relatif	8	2	8D	141

BVC : branchement si non débordement.

Opération :
PC ← (PC) + 02 + déplacement si (V) = 0

Description :
branchement si V est à 0.

Indicateurs affectés :
aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
BVC	relatif	3	2	28	040

BVS : branchement si débordement.

Opération :
PC ← (PC) + 02 + déplacement si (V) = 1

Description :
branchement si V est à 1.

Indicateurs affectés :
aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
BVS	relatif	2	2	29	041

CBA : comparaison des deux accumulateurs.

Opération :
(A) – (B)
Description :
le contenu de B est soustrait du contenu de A. Mais A et B restent inchangés. Seuls les indicateurs correspondant au résultat de cette soustraction sont modifiés.

Indicateurs affectés :

N : est mis à 1 si le résultat est négatif (bit 7 à 1), sinon remis à 0
Z : est mis à 1 si le résultat est nul, sinon remis à 0
V : mis à 1 si la soustraction provoque un débordement en complément à 2, sinon remis à 0
C : mis à 1, si en valeur absolue, le contenu de B est supérieur au contenu de A, sinon remis à 0

Expression booléenne :

$N = R_7$
$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$
$V = A_7 \cdot \overline{B_7} \cdot \overline{R_7} + \overline{A_7} \cdot B_7 \cdot R_7$
$C = \overline{A_7} \cdot B_7 + B_7 \cdot R_7 + R_7 \cdot \overline{A_7}$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes mechine	
				hexa	décimei
CBA	implicite	2	1	11	017

CLC : mise à zéro de la retenue.

Opération :
C — 0

Description .
met le bit C du registre d'état à zéro.

Indicateurs affectés :
C : mis à 0

Expression booléenne
C = 0

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
CLC	implicite	2	1	0C	012

CLI : mise à zéro du masque d'interruption.

Opération :
I ← 0

Description :
met le bit I (masque d'interruption) du registre d'état à 0. Ceci autorise le microprocesseur à prendre en compte les interruptions provenant d'un périphérique et arrivant sur la ligne IRQ (''Interrupt Request'') du 6803.

Indicateurs affectés :
I : mis à 0

Expression booléenne :
I = 0

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
CLI	implicite	2	1	0E	014

CLR : mise à zéro.

Opération :
Acc ← 0 ou M ← 0

Description
le contenu de l'un des accumulateurs, ou d'un mot mémoire est remplacé par zéro.

Indicateurs affectés
N : mis à 0
Z : mis à 1
V : mis à 0
C : mis à 0

Expression booléenne :
N = 0
Z = 1
V = 0
C = 0

Codes :

mnémoniques	edressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
CLR A	implicite	2	1	4F	079
CLR B	implicite	2	1	5F	095
CLR	étendu	6	3	7F	127
CLR	indexé	6	2	6F	111

CLV : mise à zéro de l'indicateur de débordement.

Opération :
V ← 0

Description :
le bit V du registre d'état est mis à 0.

Indicateurs affectés :
V : mis à 0

Expression booléenne :
V = 0

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
CLV	implicite	2	1	0A	010

CMP : comparaison entre accumulateur et mot mémoire

Opération :

(ACC) − (M)

Description :

comparaison du contenu de l'un des accumulateurs avec le contenu d'un mot mémoire. Les indicateurs correspondants sont modifiés en conséquence (et seront utilisés pour contrôler des branchements conditionnels). Aucun des deux opérandes n'est modifié.

Indicateurs affectés :

N : mis à 1 si le bit 7 du résultat de la soustraction doit être à 1, sinon remis à 0
Z : mis à 1 si tous les bits du résultat doivent être nuls, sinon remis à 0
V : mis à 1 si la soustraction provoque un débordement en complément à 2, sinon remis à 0
C : mis à 1 si la valeur absolue du contenu du mot mémoire est supérieur à la valeur absolue du contenu de l'accumulateur, sinon remis à 0

Expression booléenne :

$N = R_7$

$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$

$V = \overline{ACC_7} \cdot M_7 \cdot \overline{R_7} + \overline{ACC_7} \cdot M_7 \cdot \overline{R_7}$

$C = \overline{ACC_7} \cdot M_7 + M_7 \cdot R_7 + R_7 \cdot \overline{ACC_7}$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
CMP A	immédiat	2	2	81	129
CMP A	direct	3	2	91	145
CMP A	étendu	4	3	B1	177
CMP A	indexé	4	2	A1	161
CMP B	immédiat	2	2	C1	193
CMP B	direct	3	2	D1	209
CMP B	étendu	4	3	F1	241
CMP B	indexé	4	2	E1	225

COM : complémentation (à 1).

Opération :

ACC ← ($\overline{\text{ACC}}$) = 'FF − (ACC) ou M ← ($\overline{\text{M}}$) = 'FF − (M)

Description :

remplace le contenu de l'un des accumulateurs ou d'un mot mémoire par son complément à 1 (chaque bit est remplacé par son complément à 1).

Indicateurs affectés :

N : mis à 1 si le bit 7 du résultat est à 1, sinon remis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à 0
V : mis à 0
C : mis à 0

Expression booléenne :

$N = R_7$
$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$
$V = 0$
$C = 1$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
COM A	implicite	2	1	43	067
COM B	implicite	2	1	53	0B3
COM	étendu	6	3	73	115
COM	indexé	6	2	63	099

CPX : comparaison avec IX.

Opération :
(IXL) − (M + 1)
(IXH) − (M)

Description :

l'octet de poids fort du contenu du registre d'index est comparé au contenu de l'octet dont l'adresse est spécifiée dans le programme. L'octet de poids faible du contenu du registre d'index est comparé au contenu de l'octet suivant celui dont l'adresse figure dans le programme. Les indicateurs sont modifiés en fonction du résultat de cette comparaison. Aucun des deux opérandes n'est modifié.
Le bit Z, modifié par cette comparaison, pourra être utilisé comme contrôle d'un branchement inconditionnel. Les bits N et V, bien que mis à jour par cette comparaison ne pourront contrôler un branchement conditionnel.

Indicateurs affectés :

N : mis à 1 si le bit 7 du résultat de la soustraction du contenu de M par rapport à l'octet de poids fort de IX est à 1, sinon remis à 0
Z : mis à 1 si les 16 bits du résultat de la soustraction sont nuls, sinon remis à 0
V : mis à 1 si la soustraction par rapport à l'octet de poids fort de IX provoque un débordement en complément à 2, sinon remis à 0

Expression booléenne :

$N = \overline{R}_{15}$
$Z = \overline{R}_{15} \cdot \overline{R}_{14} \cdot \ldots \overline{R}_{2} \cdot \overline{R}_{1} \cdot \overline{R}_{0}$
$V = IX_{15} \cdot \overline{M}_{7} \cdot \overline{R}_{15} + \overline{IX}_{15} \cdot M_{7} \cdot R_{15}$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
CPX	immédiat	4	3	8C	140
CPX	direct	5	2	9C	156
CPX	étendu	6	3	BC	188
CPX	indexé	6	2	AC	172

DAA : ajustement décimal de l'accumulateur A.

Opération :

$$A \leftarrow (A) + \begin{cases} '00 \\ '06 \\ '60 \\ '66 \end{cases}$$

Description :

ajuste le contenu de l'accumulateur A et le bit C pour obtenir un résultat conforme à l'arithmétique DCB. Attention, ne fonctionne que si le contenu de A et l'état de C et H sont le résultat d'une addition ABA, ADC ou ADD sur des opérandes DCB.

Indicateurs affectés :

N : mis à 1 si le bit 7 du résultat est à 1, sinon remis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à 0.
V : indéfini
C : mis à 1 si le résultat est supérieur à 99

Expression booléenne :

$N = R7$
$Z = \overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
DAA	implicite	2	1	19	025

DEC : décrémentation (de 1).

Opération :

ACC ← (ACC) − 01 ou M ← (M) − 01

Description :

décrémente l'un des accumulateurs ou un mot mémoire de 1.

Indicateurs affectés :

N : mis à 1 si le bit 7 du résultat est à 1, sinon remis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à 0
V : mis à 1 s'il y a débordement en complément à deux du résultat, sinon remis à 0. Ce débordement ne se produit que si et seulement si le contenu de l'accumulateur, ou du mot mémoire, était '080 avant l'opération.

Expression booléenne :

$N = R7$
$Z = \overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$
$V = \overline{R7} \cdot R6 \cdot R5 \cdot R4 \cdot R3 \cdot R2 \cdot R1 \cdot R0$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
DEC A	implicite	2	1	4A	074
DEC B	implicite	2	1	5A	090
DEC	étendu	6	3	7A	122
DEC	indexé	6	2	6A	106

DES : décrémentation du pointeur de pile (de 1).

Opération :
SP ← (SP) − 01

Description :
retranche 1 au contenu du pointeur de pile. Résultat dans SP.

Indicateurs affectés :
aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
DES	implicite	3	1	34	052

DEX : décrémentation du registre d'index (de 1).

Opération :
IX ← (IX) − 1

Description :
retranche 1 au contenu du registre d'index. Résultat dans IX.

Indicateurs affectés :
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à 0

Expression booléenne :
$Z = \overline{R}_{15} \bullet \overline{R}_{14} \bullet \ldots \overline{R}_{2} \bullet \overline{R}_{1} \bullet \overline{R}_{0}$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
DEX	implicite	3	1	09	009

EOR : OU exclusif.

Opération :

(Acc) ← (Acc) ⊕ (M)

Description :

réalise le "OU exclusif" entre le contenu de l'un des accumulateurs et le contenu d'un mot mémoire et place le résultat dans ce même accumulateur (le OU exclusif est réalisé bit par bit).

Indicateurs affectés :

N : mis à 1 si le bit 7 du résultat est à 1, sinon remis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à 0
V : mis à 0

Expression booléenne :

$N = R_7$
$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$
$V = 0$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
EOR A	immédiat	2	2	88	136
EOR A	direct	3	2	98	152
EOR A	étendu	4	3	B8	184
EOR A	indexé	4	2	A8	168
EOR B	immédiat	2	2	C8	200
EOR B	direct	3	2	D8	216
EOR B	étendu	4	3	F8	248
EOR B	indexé	4	2	E8	232

INC : incrémentation (de 1)

Opération :

(ACC) ← (ACC) + 01 ou M ← (M) + 01

Description :

ajoute 1 au contenu de l'un des accumulateurs ou d'un mot mémoire

Indicateurs affectés :

N : mis à 1 si le bit 7 du résultat est à 1, sinon remis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à 0
V : mis à 1 s'il y a débordement en complément à deux sur le résultat. Ce débordement ne se produit que si et seulement si le contenu de l'accumulateur, ou du mot mémoire, avant incrémentation était '7F.

Expression booléenne :

$N = R_7$

$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$

$V = \overline{ACC_7} \cdot ACC_6 \cdot ACC_5 \cdot ACC_4 \cdot ACC_3 \cdot ACC_2 \cdot ACC_1 \cdot ACC_0$

ou

$V = \overline{M_7} \cdot M_6 \cdot M_5 \cdot M_4 \cdot M_3 \cdot M_2 \cdot M_1 \cdot M_0$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
INC A	implicite	2	1	4C	076
INC B	implicite	2	1	5C	092
INC	étendu	6	3	7C	124
INC	indexé	6	2	6C	108

INS : incrémentation du pointeur de pile (de 1).

Opération :
SP ← (SP) + 01

Description :
ajoute 1 au contenu du pointeur de pile.

Indicateurs affectés :
aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
INS	implicite	3	1	31	049

INX : incrémentation du registre d'index (de 1).

Opération :
IX ← (IX) + 01

Description :
ajoute 1 au contenu du registre d'index.

Indicateurs affectés :
Z : mis à 1 si les 16 bits du résultat sont nuls, sinon remis à 0

Expression booléenne :
$Z = \overline{R}_{15} \bullet \overline{R}_{14} \bullet \ldots \overline{R}_{2} \bullet \overline{R}_{1} \bullet \overline{R}_{0}$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
INX	implicite	3	1	08	008

JMP : saut à une adresse.

Opération :
PC ← adresse

Description :
un saut est effectué à l'instruction dont l'adresse est indiquée dans le programme. Cette adresse peut être absolue (mode étendu) ou indexée.

Indicateurs affectés :
aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
JMP	étendu	3	3	7E	126
JMP	indexé	3	2	6E	110

JSR : saut à un sous-programme.

Opération :

soit PC ← (PC) + 03 (si mode étendu)
soit PC ← (PC) + 02 (si mode indexé)
puis ↓ (PCL)
SP ← (SP) − 01
↓ (PCH)
SP ← (SP) − 01
PC ← adresse

Description :

l'adresse de la première instruction qui suit le JSR est emplilée (octect par octet), puis un saut est effectué à l'adresse indiquée dans le programme. Cette adresse peut être absolue (mode étendu) ou indexée.

Indicateurs affectés :

aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
JSR	étendu	6	3	BD	189
JSR	indexé	6	2	AD	173

LDA : chargement accumulateur.

Opération :

Acc ← (M)

Description :

charge une donnée dans l'un des deux accumulateurs.

Indicateurs affectés :

N : mis à 1 si le bit 7 de la donnée est à 1, sinon remis à 0
Z : mis à 1 si tous les bits de la donnée sont nuls, sinon remis à 0
V : mis à 0

Expression booléenne :

$N = R7$
$Z = \overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$
$V = 0$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
LDA A	immédiat	2	2	86	134
LDA A	direct	3	2	96	150
LDA A	étendu	4	3	B6	182
LDA A	indexé	4	2	A6	166
LDA B	immédiat	2	2	C6	198
LDA B	direct	3	2	D6	214
LDA B	étendu	4	3	F6	246
LDA B	indexé	4	2	E6	230

LDD

LDD : chargement du double accumulateur.

Opération :

A ← (M) et B ← (M+1)

Description :

charge le double accumulateur A:B avec une donnée sur 16 bits.

Indicateurs affectés :

N : mis à 1 si le bit 7 de A est à 1, sinon remis à 0
Z : mis à 1 si les 16 bits du résultat (et donc de la donnée) sont nuls, sinon remis à 0
V : mis à 0

Expression booléenne :

$N = \underline{A}7 = \underline{R}15$
$Z = \overline{R}15 \bullet \overline{R}14 \bullet \overline{R}13 \ldots \overline{R}2 \bullet \overline{R}1 \bullet \overline{R}0$
$V = 0$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
LDD	immédiat	3	3	CC	204
LDD	direct	4	2	DC	220
LDD	étendu	5	3	FC	252
LDD	indexé	5	2	EC	236

LDS : chargement du pointeur de pile.

Opération :

SPH ← (M)
SPL ← (M+1)

Description :

place dans le pointeur de pile, le contenu sur 16 bits, des mots mémoire d'adresses respectives M et M+1, où M est l'adresse spécifiée dans le programme.

Indicateurs affectés :

N : mis à 1 si le bit 15 du pointeur de pile est mis à 1 par l'opération, sinon remis à 0
Z : mis à 1 si tous les bits du pointeur de pile sont mis à 0 par l'opération, sinon remis à 0
V : mis à 0

Expression booléenne :

$N = R_{15}$
$Z = \overline{R}_{15} \bullet \overline{R}_{14} \bullet \overline{R}_{13} \ldots \overline{R}_{2} \bullet \overline{R}_{1} \bullet \overline{R}_{0}$
$V = 0$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
LDS	immédiat	3	3	8E	142
LDS	direct	4	2	9E	158
LDS	étendu	5	3	BE	190
LDS	indexé	5	2	AE	174

LDX : chargement du registre d'index.

Opération :

IXH ← (M)
IXL ← (M + 1)

Description :

charge l'octet de poids fort du registre d'index avec le contenu du mot mémoire dont l'adresse est spécifiée dans le programme et charge l'octet de poids faible du registre d'index avec le contenu du mot mémoire situé à l'adresse suivante.

Indicateurs affectés :

N : mis à 1 si le bit 15 du registre d'index est mis à 1 par l'opération de chargement, sinon remis à 0
Z : mis à 1 si tous les bits du registre d'index sont mis à 0 par l'opération de chargement, sinon remis à 0
V : mis à 0

Expression booléenne :

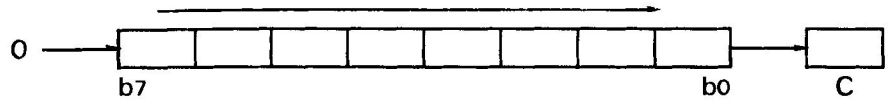
$N = R_{15}$
$Z = \overline{R}_{15} \bullet \overline{R}_{14} \bullet \overline{R}_{13} \ldots \overline{R}_{2} \bullet \overline{R}_{1} \bullet \overline{R}_{0}$
$V = 0$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
LDX	immédiat	3	3	CE	206
LDX	direct	4	2	DE	222
LDX	étendu	5	3	FE	254
LDX	indexé	5	2	EE	238

LSR : décalage logique à droite.

Opération :



Description :

décalage des bits d'un des accumulateurs ou du contenu d'un mot mémoire d'une position vers la droite. Le bit 7 reçoit la valeur 0 tandis que le bit C du registre d'état reçoit le contenu du bit 0.

Indicateurs affectés :

N : mis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à 0
V : mis à 1 si, après le décalage, soit (N est à 1 et C est à 0), soit (N est à 0 et C est à 1), sinon remis à 0
C : mis à 1 si, avant le décalage, le bit 0 de l'accumulateur ou du mot mémoire était à 1, sinon mis à 0

Expression booléenne :

$N = 0$
$Z = \overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$
$V = N \oplus C$ (N et C après opération de décalage)
$C = M0$

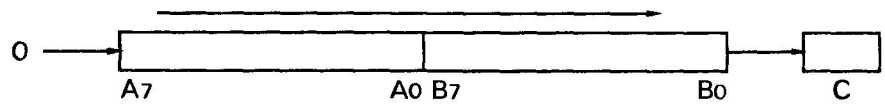
Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
LSR A	implicite	2	1	44	068
LSR B	implicite	2	1	54	084
LSR	étendu	6	3	74	116
LSR	indexé	6	2	64	100

LSRD : décalage logique à droite du double accumulateur.

Opération :

A:B ← (A:B) ÷ 2



Description :

décalage à droite d'une position du double accumulateur A:B (sur 16 bits). Le bit 7 de A est mis à 0 tandis que le bit C du registre d'état reçoit le contenu du bit 0 de B.

Indicateurs affectés :

N : mis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à 0
V : mis à 1 si, après décalage, soit (N est à 1 et C est à 0), soit (N est à 0 et C est à 1), sinon remis à 0
C : mis à 1 si, avant le décalage, le bit 0 de B était à 1, sinon mis à 0

Expression booléenne :

$N = 0$
$Z = \overline{R}_{15} \cdot \overline{R}_{14} \cdot \overline{R}_{13} \ldots \overline{R}_{2} \cdot \overline{R}_{1} \cdot \overline{R}_{0}$
$V = N \oplus C$ (N et C après opération de décalage)
$C = B_0$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
LSRD	implicite	3	1	04	004

MUL : multiplication de A et B.

Opération :

A:B ← (A) × (B)

Description :

place dans le double registre A:B (sur 16 bits), le résultat de la multiplication du contenu de A (sur 8 bits), par le contenu de B (sur 8 bits).

Indicateurs affectés :

C : mis à 1 s'il y a une retenue sur le bit le plus significatif du résultat, sinon mis à 0

Expression booléenne :

C = R7

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
MUL	implicite	10	1	3D	061

NEG : complémentation à 2.

Opération :

ACC ← − (ACC) = 00 − (ACC) ou M ← − (M) = 00 − (M)

Description :

remplace le contenu d'un des accumulateurs, ou d'un mot mémoire par son complément à 2 ; (remarque : le complément à 2 de '80 est '80).

Indicateurs affectés :

N : mis à 1 si le bit 7 du résultat est à 1, sinon remis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à 0
V : mis à 1 si le contenu de l'accumulateur ou du mot mémoire était '80, sinon remis à 0
C : mis à 0 si le contenu de l'accumulateur ou du mot mémoire était à '00 ; dans tous les autres cas, mis à 1

Expression booléenne :

$N = R7$
$Z = \overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$
$V = R7 \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$
$C = R7 + R6 + R5 + R4 + R3 + R2 + R1 + R0$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
NEG A	implicite	2	1	40	064
NEG B	implicite	2	1	50	080
NEG	étendu	6	3	70	112
NEG	indexé	6	2	60	096

NOP : pas d'opération.

Opération :
aucune

Description :
cette instruction ne fait rien. Seul le compteur ordinal est incrémenté de 1 pour pointer sur l'instruction suivante. Peut servir à boucher des trous créés par des corrections de programme ou à introduire une temporisation (2 cycles).

Indicateurs affectés :
aucun

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
NOP	implicite	2	1	01	001

ORA : OU inclusif.

Opération :

Acc ← (Acc) $\vee$ (M)

Description :

réalise le "OU" logique entre le contenu de l'un des accumulateurs et le contenu d'un mot mémoire. Le résultat de l'opération est placé dans ce même accumulateur.

Indicateurs affectés :

N : mis à 1 si le bit 7 du résultat est à 1, sinon remis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à 0
V : mis à 0

Expression booléenne :

$N = R_7$
$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$
$V = 0$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
ORA A	immédiat	2	2	8A	138
ORA A	direct	3	2	9A	154
ORA A	étendu	4	3	BA	186
ORA A	indexé	4	2	AA	170
ORA B	immédiat	2	2	CA	202
ORA B	direct	3	2	DA	218
ORA B	étendu	4	3	FA	250
ORA B	indexé	4	2	EA	234

PSH : empiler accumulateur.

Opération :

↓ (ACC)
SP ← (SP) − 01

Description :

le contenu de l'un des accumulateurs est empilé à l'adresse contenue dans le pointeur de pile puis celui-ci est décrémenté de 1.

Indicateurs affectés :

aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
PSH A	implicite	3	1	36	054
PSH B	implicite	3	1	37	055

PSHX : empiler le registre d'index.

Opération :

↓ (IXL)
SP ← (SP) − 01
↓ (IXH)
SP ← (SP) − 01

Description :

le contenu du registre d'index est empilé à l'adresse contenue dans le pointeur de pile (on empile d'abord l'octet de poids faible de IX, puis l'octet de poids fort). Le pointeur de pile est décrémenté de 2.

Indicateurs affectés :

aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
PSH X	implicite	4	1	3C	060

PUL : dépiler accumulateur.

Opération :

SP ← (SP) + 01
↑ Acc

Description :

le pointeur de pile est incrémenté de 1. L'accumulateur reçoit le contenu du mot-mémoire dont l'adrese est contenue dans le pointeur de pile.

Indicateurs affectés :

aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
PUL A	implicite	4	1	32	050
PUL B	implicite	4	1	33	051

PULX : dépiler le registre d'index.

Opération :

SP ← (SP) + 01
↑ IXH
SP ← (SP) + 01
↑ IXL

Description :

le pointeur de pile est incrémenté. IX reçoit le contenu des deux octets en sommet de pile.

Indicateurs affectés :

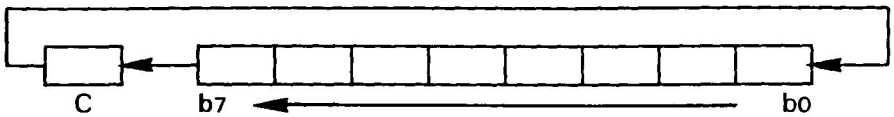
aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
PUL X	implicite	5	1	38	056

ROL : rotation à gauche.

Opération :



Description :

décalage du contenu de l'accumulateur (ou d'un mot mémoire) d'une position vers la gauche. Le bit 0 est chargé par le bit C du registre d'état, alors que ce même bit C reçoit la valeur du bit 7.

Indicateurs affectés :

N : mis à 1 si le bit 7 du résultat est à 1, sinon remis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à 0
V : mis à 1 si, après la rotation, soit (N est à 1 et C est à 0), soit (N est à 0 et C est à 1), sinon remis à 0
C : mis à 1 si, avant la rotation, le bit 7 de l'accumulateur ou du mot mémoire était à 1, sinon mis à 0

Expression booléenne :

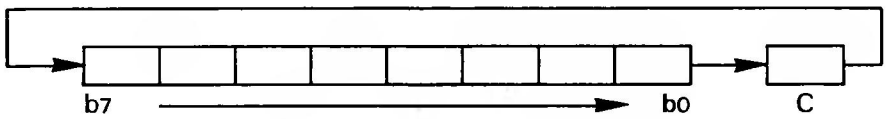
$N = R_7$
$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$
$V = N \oplus C$ (N et C après rotation)
$C = ACC_7$ ou M_7

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
ROL A	implicite	2	1	49	073
ROL B	implicite	2	1	59	089
ROL	étendu	6	3	79	121
ROL	indexé	6	2	69	105

ROR : rotation à droite.

Opération :



Description :

décalage du contenu de l'accumulateur (ou d'un mot mémoire) d'une position vers la droite. Le bit 7 reçoit le contenu du bit C du registre d'état, tandis que ce bit C reçoit le contenu du bit 0.

Indicateurs affectés :

N : mis à 1 si le bit 7 du résultat est à 1, sinon remis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à 0
V : mis à 1 si, après la rotation, soit (N est à 1 et C est à 0), soit (N est à 0 et C est à 1) ; sinon remis à 0
C : mis à 1 si, avant la rotation, le bit 0 de l'accumulateur ou du mot mémoire était à 1, sinon remis à 0

Expression booléenne :

$N = R_7$
$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$
$V = N \oplus C$ (N et C avant rotation)
$C = Acc_0$ ou M_0

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
ROR A	implicite	2	1	46	070
ROR B	implicite	2	1	56	086
ROR	étendu	6	3	76	118
ROR	indexé	6	2	66	102

RTI : retour d'une interruption.

Opération :

SP ← (SP) + 01 , ↑ CC
SP ← (SP) + 01 , ↑ B
SP ← (SP) + 01 , ↑ A
SP ← (SP) + 01 , ↑ IXH
SP ← (SP) + 01 , ↑ IXL
SP ← (SP) + 01 , ↑ PCH
SP ← (SP) + 01 , ↑ PCL

Description :

restaure les contenus du registre d'état, des accumulateurs A et B, du registre d'index et du compteur ordinal qui avaient étés sauvés dans la pile et remet à jour le pointeur de pile.

Indicateurs affectés :

tous ; restaurés en l'état stocké dans la pile.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
RTI	implicite	10	1	3B	059

RTS : retour d'un sous-programme.

Opération :

SP ← (SP) + 01 , ↑ PCH
SP ← (SP) + 01 , ↑ PCL

Description :

le pointeur de pile est incrémenté de 1 et l'octet de poids fort du compteur ordinal est restauré à partir de l'octet se trouvant en sommet de pile. Puis le pointeur de pile est de nouveau incrémenté de 1 et c'est l'octet de poids faible du compteur ordinal qui est restauré à partir du sommet de la pile.

Indicateurs affectés :

aucun.

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
RTS	implicite	5	1	39	057

SBA : soustraction des accumulateurs.

Opération :
A ← (A) − (B)
Description :
retranche le contenu de l'accumulateur B du contenu de l'accumulateur A. Le résultat est placé dans A. Le contenu de B n'est pas modifié.

Indicateurs affectés :
N : mis à 1 si le bit 7 du résultat est à 1, sinon remis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à 0
V : mis à 1 s'il y a un débordement en complément à 2 sur le résultat de l'opération, sinon remis à 0
C : mis à 1, si la valeur absolue du contenu de B, plus la retenue précédente, est supérieure à la valeur absolue du contenu de A, sinon remis à 0

Expression booléenne :
$N = R_7$
$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$
$V = A_7 \cdot \overline{B_7} \cdot R_7 + \overline{A_7} \cdot B_7 \cdot \overline{R_7}$
$C = \overline{A_7} \cdot B_7 + B_7 \cdot R_7 + R_7 \cdot \overline{A_7}$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
SBA	implicite	2	1	10	016

SBC : soustraction avec retenue.

Opération :

ACC ← (ACC) − (M) − C

Description :

retranche le contenu d'un mot mémoire et le bit C au contenu de l'accumulateur et place le résultat dans ce même accumulateur.

Indicateurs affectés :

N : mis à 1 si le bit 7 du résultat est à 1, sinon remis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à 0
V : mis à 1 s'il y a débordement en complément à 2 sur le résultat, sinon remis à 0
C : mis à 1 si la valeur absolue du contenu du mot mémoire plus le bit C est supérieure au contenu en valeur absolue de l'accumulateur, sinon remis à 0

Expression booléenne :

$N = R_7$
$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$
$V = \underline{ACC_7} \cdot \overline{M_7} \cdot \overline{R_7} + \overline{ACC_7} \cdot M_7 \cdot \underline{R_7}$
$C = \overline{ACC_7} \cdot M_7 + M_7 \cdot R_7 + R_7 \cdot \overline{ACC_7}$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
SBC A	immédiat	2	2	82	130
SBC A	direct	3	2	92	146
SBC A	étendu	4	3	B2	178
SBC A	indexé	4	2	A2	162
SBC B	immédiat	2	2	C2	194
SBC B	direct	3	2	D2	210
SBC B	étendu	4	3	F2	242
SBC B	indexé	4	2	E2	226

SEC : mise à 1 de la retenue.

Opération :
C ← 1

Description :
met à 1 le bit C du registre d'état.

Indicateurs affectés :
C : mis à 1

Expression booléenne :
C = 1

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
SEC	implicite	2	1	0D	013

SEI : mise à 1 du masque d'interruption.

Opération :

I ← 1

Description :

met à 1 le bit I du registre d'état. Toutes les interruptions en provenance des périphériques sont inhibées et le 6803 continuera à exécuter des instructions de programme sans pouvoir être interrompu tant que le bit I n'aura pas été remis à 0 (utilisé par exemple au début de la routine de sauvegarde ou de chargement avec une cassette).

Indicateurs affectés :

I : mis à 1

Expression booléenne :

I = 1

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
SEI	implicite	2	1	0F	015

SEV : mise à 1 du bit de débordement.

Opération :
V ← 1

Description :
met à 1 le bit V du registre d'état.

Indicateurs affectés :
V : mis à 1

Expression booléenne
V = 1

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
SEV	implicite	2	1	0B	011

STA : rangement en mémoire d'un accumulateur.

Opération :

$M \leftarrow (Acc)$

Description :

recopie le contenu de l'accumulateur en mémoire, à l'adresse spécifiée par le programme. Le contenu de l'accumulateur reste inchangé.

Indicateurs affectés :

N : mis à 1 si le bit 7 de l'accumulateur est à 1, sinon remis à 0
Z : mis à 1 si tous les bits de l'accumulateur sont nuls, sinon remis à 0
V : mis à 0

Expression booléenne :

$N = Acc7$
$Z = \overline{Acc7} \cdot \overline{Acc6} \cdot \overline{Acc5} \cdot \overline{Acc4} \cdot \overline{Acc3} \cdot \overline{Acc2} \cdot \overline{Acc1} \cdot \overline{Acc0}$
$V = 0$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
STA A	direct	3	2	97	151
STA A	étendu	4	3	B7	183
STA A	indexé	4	2	A7	167
STA B	direct	3	2	D7	215
STA B	étendu	4	3	F7	247
STA B	indexé	4	2	E7	231

STD

STAD : rangement en mémoire du double accumulateur.

Opération :

M:M+1 ← (A:B)

Description :

recopie le contenu du double accumulateur (sur 16 bits) en mémoire ; à l'adresse spécifiée par le programme, on recopie le contenu de A et à l'adresse suivante le contenu de B. Les contenus des accumulateurs A et B ne sont pas modifiés.

Indicateurs affectés :

N : mis à 1 si le bit 7 de l'accumulateur A est à 1, sinon remis à 0
Z : mis à 1 si tous les bits du double accumulateur sont nuls, sinon remis à 0
V : mis à 0

Expression booléenne :

$N = A_7$
$Z = \overline{A}_7 \cdot \overline{A}_6 \cdot \overline{A}_5 \cdot \overline{A}_4 \cdot \overline{A}_3 \cdot \overline{A}_2 \cdot \overline{A}_1 \cdot \overline{A}_0 \cdot \overline{B}_7 \cdot \overline{B}_6 \cdot \overline{B}_5 \cdot \overline{B}_4 \cdot \overline{B}_3 \cdot \overline{B}_2 \cdot \overline{B}_1 \cdot \overline{B}_0$
$V = 0$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
STAD	direct	4	2	DD	221
STAD	étendu	5	3	FD	253
STAD	indexé	5	2	ED	237

STS : rangement en mémoire du pointeur de pile.

Opération :

M ← (SPH)
M+1 ←(SPL)

Description :

recopie l'octet de poids fort du pointeur de pile à l'adresse spécifiée par le programme et recopie l'octet de poids faible à l'adresse suivante, c'est-à-dire à l'adresse spécifiée par le programme plus un.

Indicateurs affectés :

N : mis à 1 si le bit 15 du pointeur de pile est à 1, sinon remis à 0
Z : mis à 1 si tous les bits du pointeur de pile sont nuls, sinon remis à 0
V : mis à 0

Expression booléenne :

$N = \overline{SP_{15}}$
$Z = \overline{SP_{15}} \cdot \overline{SP_{14}} \cdot \overline{SP_{13}} \ldots \overline{SP_2} \cdot \overline{SP_1} \cdot \overline{SP_0}$
$V = 0$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
STS	direct	4	2	9F	159
STS	étendu	5	3	BF	191
STS	indexé	5	2	AF	175

STX : rangement en mémoire du registre d'index.

Opération :

M ← (IXH)
M+1 ←(IXL)

Description :

recopie l'octet de poids fort du registre d'index l'adresse spécifiée par le programme et recopie l'octet de poids faible à l'adresse suivante, c'est-à-dire à l'adresse spécifiée par le programme plus 1.

Indicateurs affectés :

N : mis à 1 si le bit 15 du registre d'index est à 1, sinon remis à 0
Z : mis à 1 si tous les bits du registre d'index sont nuls, sinon remis à 0
V : mis à 0

Expression booléenne :

$N = IX_{15}$
$Z = \overline{IX_{15}} \cdot \overline{IX_{14}} \cdot \overline{IX_{13}} \ldots \overline{IX_{2}} \cdot \overline{IX_{1}} \cdot \overline{IX_{0}}$
$V = 0$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
STX	direct	4	2	DF	223
STX	étendu	5	3	FF	255
STX	indexé	5	2	EF	239

SUB : soustraction sur 8 bits.

Opération :

ACC ← (ACC) − (M)

Description :

retranche le contenu d'un mot mémoire au contenu de l'accumulateur et place le résultat dans ce même accumulateur.

Indicateurs affectés :

N : mis à 1 si le bit 7 du résultat est à 1, sinon remis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à 0
V : mis à 1 s'il y a débordement en complément à deux sur le résultat, sinon remis à 0
C : mis à 1 si la valeur absolue du contenu du mot mémoire est supérieure au contenu en valeur absolue de l'accumulateur, sinon remis à 0

Expression booléenne :

$N = R_7$
$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$
$V = ACC_7 \cdot M_7 \cdot \overline{R_7} + \overline{ACC_7} \cdot M_7 \cdot R_7$
$C = \overline{ACC_7} \cdot M_7 + M_7 \cdot R_7 + R_7 \cdot \overline{ACC_7}$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
SUB A	immédiat	2	1	80	128
SUB A	direct	3	1	90	144
SUB A	étendu	4	3	B0	176
SUB A	indexé	4	2	A0	160
SUB B	immédiat	2	2	C0	192
SUB B	direct	3	2	D0	208
SUB B	étendu	4	3	F0	240
SUB B	indexé	4	2	E0	224

SUBD : soustraction sur 16 bits.

Opération :

A:B ← (A:B) − (M:M+1)

Description :

retranche au contenu du double accumulateur (sur 16 bits), le contenu des deux mots mémoires d'adresses respectives M et M+1, où M est l'adresse spécifiée par le programme, et place le résultat sur 16 bits dans ce double accumulateur.

Indicateurs affectés :

N : mis à 1 si le bit 15 du résultat est à 1, sinon remis à 0
Z : mis à 1 si tous les bits du résultat sont nuls, sinon remis à 0
V : mis à 1 s'il y a débordement en complément à deux sur le résultat, sinon remis à 0
C : mis à 1 si la valeur absolue du contenu mémoire est supérieure à la valeur absolue du contenu du double accumulateur, sinon remis à 0

Expression booléenne :

$N = R_{15}$
$Z = \overline{R}_{15} \cdot \overline{R}_{14} \cdot \overline{R}_{13} \ldots \overline{R}_{2} \cdot \overline{R}_{1} \cdot \overline{R}_{0}$
$V = A_7 \cdot \overline{M}_7 \cdot \overline{R}_{15} + \overline{A}_7 \cdot M_7 \cdot R_{15}$
$C = \overline{A}_7 \cdot M_7 + M_7 \cdot R_{15} + R_7 \cdot \overline{A}_7$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
SUBD	immédiat	4	3	83	131
SUBD	direct	5	2	93	147
SUBD	étendu	6	3	B3	179
SUBD	indexé	6	2	A3	163

SWI : interruption software.

Opération :

PC ← (PC) + 01
↓ (PCL) , SP ← (SP) − 01
↓ (PCH) , SP ← (SP) − 01
↓ (IXL) , SP ← (SP) − 01
↓ (IXH) , SP ← (SP) − 01
↓ (A) , SP ← (SP) − 01
↓ (B) , SP ← (SP) − 01
↓ (CC) , SP ← (SP) − 01
I ← 1
PCH ← ('FFFA)
PCL ← ('FFFB)

Description :

le compteur ordinal est incrémenté de 1 puis le compteur ordinal, le registre d'index, les accumulateurs A et B sont empilés. Le registre d'état est alors empilé de telle sorte que les bits H, I, N, Z, V et C sont placés respectivement dans les bits 5 à 0. Les bits 6 et 7 sont mis à 1. Le pointeur de pile est mis à jour après chaque opération. Le masque d'interruption est mis à 1 puis on charge le compteur ordinal avec le vecteur de branchement SWI placé aux adresses 'FFFA et 'FFFB.

Indicateurs effectés :

I : mis à 1

Expression booléenne :

I = 1

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
SWI	implicite	12	1	3F	063

TAB : transfert de l'accumulateur A dans l'accumulateur B.

Opération :
B ← (A)
Description :
recopie dans l'accumulateur B, le contenu de l'accumulateur A. Le contenu de A n'est pas modifié.

Indicateurs affectés :
N : mis à 1 si le bit 7 du contenu de l'accumulateur est à 1, sinon remis à 0
Z : mis à 1 si tous les bits de A sont nuls, sinon remis à 0
V : mis à 0

Expression booléenne :
$N = A_7$
$Z = \overline{A}_7 \bullet \overline{A}_6 \bullet \overline{A}_5 \bullet \overline{A}_4 \bullet \overline{A}_3 \bullet \overline{A}_2 \bullet \overline{A}_1 \bullet \overline{A}_0$
$V = 0$

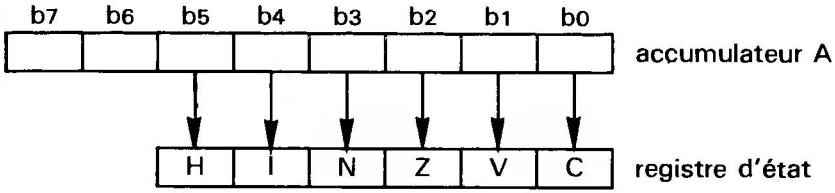
Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
TAB	implicite	2	1	16	022

TAP : recopie de A dans le registre d'état.

Opération :
CC ← (A)

Description :
recopie des bits 0 à 5 de l'accumulateur A dans les bits correspondants du registre d'état. Le contenu du registre A n'est pas modifié.



Indicateurs affectés :
tous

Expression booléenne :
H = A_5
I = A_4
N = A_3
Z = A_2
V = A_1
C = A_0

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
TAP	implicite	2	1	06	006

TBA : transfert de l'accumulateur B dans l'accumulateur A.

Opération :
A ← (B)
Description :
recopie dans l'accumulateur A du contenu de l'accumulateur B. Le contenu de B n'est pas modifié.

Indicateurs affectés :

N : mis à 1 si le bit 7 de l'accumulateur est à 1, sinon remis à 0
Z : mis à 1 si tous les bits de B sont nuls, sinon remis à 0
V : mis à 0

Expression booléenne :

$N = B_7$
$Z = \overline{B_7} \cdot \overline{B_6} \cdot \overline{B_5} \cdot \overline{B_4} \cdot \overline{B_3} \cdot \overline{B_2} \cdot \overline{B_1} \cdot \overline{B_0}$
$V = 0$

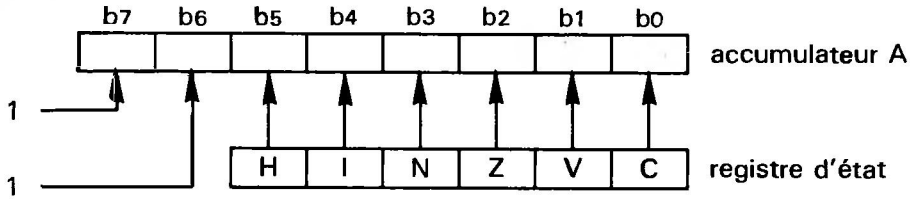
Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
TBA	implicite	2	1	17	023

TPA : recopie du registre d'état dans A.

Opération
A ← (CC)

Description
recopie des bits du registre d'état dans les bits 0 à 5 de l'accumulateur A. Les bits 6 et 7 de A sont mis à 1 et le contenu du registre d'état n'est pas modifié.



Indicateurs affectés :
aucun

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
TPA	implicite	2	1	07	007

TST : test.

Opération :

(Acc) − 00 ou (M) − 00

Description :

comparaison du contenu de l'accumulateur ou du mot mémoire avec 0 et mise à jour des bits N et Z en fonction du résultat de la comparaison.

Indicateurs affectés :

N : mis à 1 si le bit 7 de l'accumulateur ou du mot mémoire est à 1, sinon remis à 0
Z : mis à 1 si tous les bits de l'accumulateur ou du mot mémoire sont nuls, sinon remis à 0
V : mis à 0
C : mis à 0

Expression booléenne :

$N = Acc_7$
ou
$N = M_7$
$Z = \overline{Acc_7} \cdot \overline{Acc_6} \cdot \overline{Acc_5} \cdot \overline{Acc_4} \cdot \overline{Acc_3} \cdot \overline{Acc_2} \cdot \overline{Acc_1} \cdot \overline{Acc_0}$
ou
$Z = \overline{M}_7 \cdot \overline{M}_6 \cdot \overline{M}_5 \cdot \overline{M}_4 \cdot \overline{M}_3 \cdot \overline{M}_2 \cdot \overline{M}_1 \cdot \overline{M}_0$
$V = 0$
$C = 0$

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
TST A	implicite	2	1	4D	077
TST B	implicite	2	1	5D	093
TST	étendu	6	3	7D	125
TST	indexé	6	2	6D	109

TSX : transfert du pointeur de pile dans le registre d'index.

Opération :
IX ← (SP) + 01

Description :
charge le registre d'index avec le contenu du pointeur de pile augmenté de 1. Le contenu du pointeur de pile n'est pas modifié.

Indicateurs affectés :
aucun

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
TSX	implicite	3	1	30	048

TXS : transfert du registre d'index dans le pointeur de pile.

Opération :
SP ← (IX) − 01

Description :
charge le pointeur de pile avec le contenu du registre d'index diminué de 01. Le contenu du registre d'index n'est pas modifié.

Indicateurs affectés :
aucun

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
TXS	implicite	3	1	35	053

WAI : attente d'interruption

Opération :

PC ← (PC) + 01
↓ (PCL) , SP ← (SP) − 01
↓ (PCH) , SP ← (SP) − 01
↓ (IXL) , SP ← (SP) − 01
↓ (IXH) , SP ← (SP) − 01
↓ (A) , SP ← (SP) − 01
↓ (B) , SP ← (SP) − 01
↓ (RE) , SP ← (SP) − 01

Description :

le compteur ordinal est incrémenté de 1 puis les contenus du compteur ordinal, du registre d'index, des accumulateurs A et B et du registre d'état sont empilés. Le pointeur de pile est mis à jour. Puis l'exécution du programme est interrompue en attente d'une interruption, sur la ligne IRQ, en provenance d'un périphérique. Lorsque se produit cette interruption, le bit I est mis à 1 (s'il était à 0) et le compteur ordinal reçoit le contenu du vecteur de branchement situé aux adresses 'FFF8 et 'FFF9 (IRQ1).

Indicateurs affectés :

I : si I était à 0, lorsqu'une interruption est signalée, I est alors mis à 1

Codes :

mnémoniques	adressage	nbre cycles	nbre octets	codes machine	
				hexa	décimal
WAI	implicite	9	1	3E	062

TABLE DES CODES OPÉRATION DU MC 6803

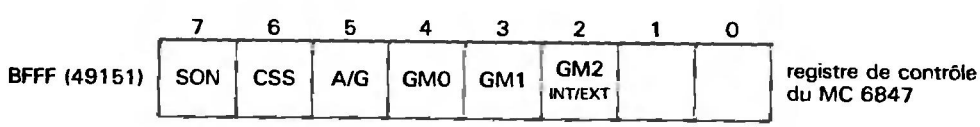
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	–	NOP inhr			LSRD* inhr	ASLD* inhr	TAP inhr	TPA inhr	INX inhr	DEX inhr	CLV inhr	SEV inhr	CLC inhr	SEC inhr	CLI inhr	SEI inhr
1	SBA inhr	CBA inhr	–	–	–	–	TAB inhr	TBA inhr	–	DAA inhr	–	ABA inhr	–	–	–	–
2	BRA rel	BRN rel	BHI rel	BLS rel	BCC rel	BCS rel	BNE rel	BEQ rel	BVC rel	BVS rel	BPL rel	BMI rel	BGE rel	BLT rel	BGT rel	BLE rel
3	TSX inhr	INS inhr	PUL A inhr	PUL B inhr	DES inhr	TXS inhr	PSH A inhr	PSH B inhr	PUL X* inhr	RTS inhr	ABX* inhr	RTI inhr	PSH X* inhr	MUL* inhr	WAI inhr	SWI inhr
4	NEG A inhr	–	–	COM A inhr	LSR A inhr	–	ROR A inhr	ASR A inhr	ASL A inhr	ROL A inhr	DEC A inhr	–	INC A inhr	TST A inhr	–	CLR A inhr
5	NEG B inhr	–	–	COM B inhr	LSR B inhr	–	ROR B inhr	ASR B inhr	ASL B inhr	ROL B inhr	DEC B inhr	–	INC B inhr	TST B inhr	–	CLR B inhr
6	NEG ind	–	–	COM ind	LSR ind	–	ROR ind	ASR ind	ASL ind	ROL ind	DEC ind	–	INC ind	TST ind	JMP ind	CLR ind
7	NEG ext	–	–	COM ext	LSR ext	–	ROR ext	ASR ext	ASL ext	ROL ext	DEC ext	–	INC ext	TST ext	JMP ext	CLR ext
8	SUB A imm	CMP A imm	SBC A imm	SUB D* imm	AND A imm	BIT A imm	LDA A imm	–	EOR A imm	ADC A imm	ORA A imm	ADD A imm	CPX imm	BSR rel	LDS imm	–
9	SUB A dir	CMP A dir	SBC A dir	SUB D* dir	AND A dir	BIT A dir	LDA A dir	STA A dir	EOR A dir	ADC A dir	ORA A dir	ADD A dir	CPX dir	JSR* dir	LDS dir	STS dir
A	SUB A ind	CMP A ind	SBC A ind	SUB D* ind	AND A ind	BIT A ind	LDA A ind	STA A ind	EOR A ind	ADC A ind	ORA A ind	ADD A ind	CPX ind	JSR ind	LDS ind	STS ind
B	SUB A ext	CMP A ext	SBC A ext	SUB D* ext	AND A ext	BIT A ext	LDA A ext	STA A ext	EOR A ext	ADC A ext	ORA A ext	ADD A ext	CPX ext	JSR ext	LDS ext	STS ext
C	SUB B imm	CMP B imm	SBC B imm	ADD D* imm	AND B imm	BIT B imm	LDA B imm	–	EOR B imm	ADC B imm	ORA B imm	ADD B imm	LDD* imm	–	LDX imm	–
D	SUB B dir	CMP B dir	SBC B dir	ADD D* dir	AND B dir	BIT B dir	LDA B dir	STA B dir	EOR B dir	ADC B dir	ORA B dir	ADD B dir	LDD* dir	STD* dir	LDX dir	STX dir
E	SUB B ind	CMP B ind	SBC B ind	ADD D* ind	AND B ind	BIT B ind	LDA B ind	STA B ind	EOR B ind	ADC B ind	ORA B ind	ADD B ind	LDD* ind	STD* ind	LDX ind	STX ind
F	SUB B ext	CMP B ext	SBC B ext	ADD D* ext	AND B ext	BIT B ext	LDA B ext	STA B ext	EOR B ext	ADC B ext	ORA B ext	ADD B ext	LDD* ext	STD* ext	LDX ext	STX ext

LE GÉNÉRATEUR D'ÉCRAN MC 6847

Une des particularités d'ALICE et du TANDY MC10 est d'être dotés d'un générateur-contrôleur d'écran. Ainsi, ce n'est pas le microprocesseur qui gère l'affichage sur l'écran, mais un circuit spécialisé, le MC 6847. Il s'agit d'un circuit intégré N-MOS encapsulé dans un boîtier de 40 broches (voir schéma), qui constitue l'interface entre le MC 6803 (et en général tous les microprocesseurs de la famille 6800 et 68000) et un téléviseur couleur ou noir et blanc.
Le générateur d'écran lit les données dans une mémoire et produit un signal vidéo-composite qui permettra de générer des caractères alphanumériques, mais aussi des caractères graphiques sur écran vidéo.
Une caractéristique du 6847 est d'être compatible avec le circuit intégré modulateur MC 1372 de MOTOROLA. Le signal ainsi modulé peut alors entrer sur la prise antenne soit d'un téléviseur couleur, soit d'un téléviseur noir et blanc.
Le MC 6847 peut générer des caractères alphanumériques en quatre modes et 8 modes de caractères graphiques. Toutefois, sur ALICE et MC10, il est à noter que si les quatre modes alphanumériques sont accessibles, ainsi que deux modes semi-graphiques, il est plus difficile de travailler en mode graphique, accessible uniquement depuis le langage machine. En effet, les modes graphiques disponibles sur le 6847 nécessitent au minimum 1 k de mémoire d'écran. Or la RAM est initialisée avec 512 octets de mémoire d'écran, suivis immédiatement d'une zone utilisée par le Basic (buffer d'Entrée/Sortie, tampon clavier, code source Basic, etc...). L'utilisation du Basic interdit donc le passage en mode graphique. Cependant, ces différents modes graphiques demeurent accessibles à partir de programmes écrits en langage machine ainsi que nous allons le voir avec quelques exemples.

LES MODES ALPHANUMÉRIQUES

En mode alphanumérique, les pages émises contiennent 16 lignes de 32 caractères chacune, chaque caractère étant représenté par une matrice de 8×12 pixels. Il est à noter que ces pixels ne sont pas accessibles individuellement à l'utilisateur. Le générateur de caractères est placé dans une ROM interne au 6847 et contient 64 caractères ASCII dans un format standard 5×7 pixels.
Le bus de données du 6847, sur huits bits, sera utilisé pour envoyer le code ASCII du caractère à afficher. Les deux bits restants, non utilisés, pourront servir, comme c'est le cas sur ALICE et MC10, à sélectionner le type (alphanumérique ou semi-graphique) ou le mode (normal ou inverse).
Une mémoire d'écran de 512 octets sera alors nécessaire pour l'affichage.
Un multiplexeur interne au 6847 permet une alternance à ces deux modes alphanumérique utilisant la ROM interne : il est en effet possible d'utiliser en conjugaison avec le 6847 un générateur externe de caractères, ce qui peut permettre par exemple d'étendre la police de caractère à des formes simples.
Huit broches du 6847 permettent de sélectionner le mode désiré pour l'affichage. Ces huit broches sont accessibles individuellement sur ALICE par l'adresse BFFF hexa (49151 décimal) en RAM : selon le schéma suivant :



Pour sélectionner l'un des modes d'affichage, il suffit alors de placer à l'adresse BFFF la valeur correspondante. Le tableau donné ci-après vous indique les codes correspondant à chaque mode d'affichage.
En ce qui concerne le choix de l'affichage en inverse vidéo, on constate que ce choix n'est pas effectué par l'adresse BFFF. En effet, il est sélectionné grâce à une autre adresse : l'adresse 421C hexa (ou 16924 décimal). Ce mot mémoire est initialisé par le système à la valeur 255, mais peut contenir toute valeur comprise entre 1 et 255 en mode normal. Pour passer en mode inverse, il suffit de placer 0 à cette adresse. On pourra d'ailleurs remarquer que l'action de shift/0 est de placer le complément du contenu de 421C en 421C. Ainsi lorsque 421C contient la valeur 255, une pression sur shift/0 transformera le contenu de 421C en "0" (et réciproquement). Cependant, on vient de voir, qu'en mode normal, le mot 421C peut prendre toute valeur à l'exeption de 0 et que l'action de shift/0 était de complémenter le contenu de cette adresse. Ainsi, si par exemple on a placé à l'adresse 421C la valeur 128 (mode normal), une pression sur shift/0, transformera ce contenu en 127 (complément à 255 de 128), qui correspond toujours au mode normal. Lorsque le mode inverse aura été sélectionné, ALICE placera 0 dans le bit 6 du bus de donnée du 6847.

LES MODES SEMI-GRAPHIQUES

La broche A/S du 6847, permettant de sélectionner un mode semi-graphique, est reliée au bit 7 du bus de données. Ainsi pour avoir un caractère semi-graphique affiché à l'écran faut-il envoyer sur le bus de données une valeur supérieure à 128 (bit 7 ayant alors la valeur "1").

Il existe deux modes semi-graphiques :
- le mode semi-graphique à 4 pavés
- le mode semi-graphique à 6 pavés.

Ces deux modes nécessitent 512 octets de mémoire d'écran et sont donc directement accessibles depuis le Basic. Pour les atteindre, il faut obligatoirement que le bit 5 de l'adresse BFFF soit à 0 et c'est le bit 2 de cette même adresse qui permettra de passer de l'un à l'autre.

• ***Mode semi-graphique à 4 éléments :*** c'est le mode décrit dans le manuel livré avec votre ordinateur. Il correspond à un contenu de l'adresse BFFF tel que les bits 5 et 2 soient tous les deux à "0". Le principe est le suivant : chaque caractère semi-graphique peut être défini par ses quatre éléments décrits séparément :

L_3	L_2
L_1	L_0

La valeur à envoyer sur le bus de données aura alors le format suivant

7	6	5	4	3	2	1	0
1	C_2	C_1	C_0	L_3	L_2	L_1	L_0

- Les bits 0 à 3 servent à sélectionner la luminance des éléments du caractère semi-graphique : bit à 0 correspond à élément éteint (noir)
  bit à 1 correspond à élément allumé.
- Les bits 3, 4 et 5 permettent de sélectionner la couleur des éléments allumés suivant les codes :

C_2	C_1	C_0	COULEUR
	0	0	
	0	1	
	1	0	
	1	1	
	0	0	
	0	1	
	1	0	
	1	1	

- Le bit 7, pour sa part doit toujours être à 1 car il est connecté à la broche A/S du 6847 et c'est lui qui permet de sélectionner le mode semi-graphique (lorsqu'il est à 1) ou le mode alphanumérique (lorsqu'il est à 0).

Il est à noter que c'est le seul mode d'affichage où le bit 6 de BFFF ne joue aucun rôle sur les couleurs des caractères affichés, mais intervient cependant sur la couleurs du fond.

• ***Mode semi-graphique à 6 éléments :*** dans ce mode semi-graphique il est possible d'adresser 64 éléments par ligne d'affichage et 48 éléments par colonne sur l'écran. Cependant cet affichage s'effectuera sous la forme de pavés semi-graphiques de 3 × 2 éléments. Ce mode semi-graphique sera obtenu en mettant le bit 2 de BFFF à "1" et le bit 5 à "0". Quant au bit 6, selon qu'on le mettra à "0" ou à "1" on pourra sélectionner l'un ou l'autre de deux jeux de deux couleurs. Pour atteindre ce mode semi-graphique, il est nécessaire d'envoyer un "1" sur le bit 7 du bus de données ou, en d'autre termes, d'envoyer sur le bus de données un code supéreur ou égal à 128. Nous avons là encore une limite dûe à la conception d'ALICE. En fait, le bit A/S qui est sur ALICE le bit 8 du bus de données, devrait être en principe l'un des bits de l'adresse BFFF, ce qui permettrait d'afficher ces pavés graphiques en quatre couleurs et non en deux couleurs comme c'est le cas ici. Ceci est d'autant plus curieux que les bits 0 et 1 du mot d'adresse BFFF sont inutilisés et qu'on pourrait fort bien avoir connecté l'un d'eux à la broche A/S du 6847.
Mais, examinons le fonctionnement de ce mode semi-graphique. Les pavés affichés ont le format suivant :

L_5	L_4
L_3	L_2
L_1	L_0

On devra alors envoyer sur le bus de données un code dans le format :

7	6	5	4	3	2	1	0
1	C	L_5	L_4	L_3	L_2	L_1	L_0

Comme dans le cas du mode graphique à quatre éléments, les bits 0 à 5, selon qu'ils sont à 0 ou à 1, indiquent que l'élément correspondant sera allumé ou éteint. Quant au bit 6, il indique la couleur des élément allumés :
- si le bit 6 de BFFF est à 0 et bit 6 du bus de données à 1 : ROUGE
  bit 6 du bus de données à 0 : BLEU
- si le bit 6 de BFFF est à 1 et bit 6 du bus de données à 1 : ORANGE
  bit 6 du bus de données à 0 : MAUVE
  0 sur bus de données : ROUGE

LES MODES GRAPHIQUES

Il y a huit modes haute résolution graphique disponibles sur le 6847. Il est possible d'accèder à ces modes graphiques sur ALICE, moyennant quelques précautions. En effet, tous ces modes haute résolution nécessitent une mémoire d'écran dont la taille varie de 1 k à 6 k. Aussi, certains de ces modes graphiques nécessiteront-ils impérativement

l'extension mémoire 16 k. D'autre part, on peut s'étonner de ce que la RAM système soit placée immédiatement après les 512 octets de la mémoire d'écran utilisée en mode alphanumérique, ce qui interdit l'accès à toute haute résolution à partir du Basic, car le Basic utilise cette zone pour placer toute ses variables. Aussi, l'utilisation du langage machine s'impose-t-elle dans ce cas. Cependant, cela posera malgré tout quelques problèmes, notamment pour l'emploi du clavier. En effet, la zone utilisée par ce dernier se situe de l'adresse décimale 16945 à 16954, c'est-à-dire en plein dans la zone écran haute résolution. Aussi, sera-t-il difficile de réaliser des jeux, par exemple, utilisant ces modes graphiques.
Nous allons cependant étudier rapidement ces possibilités d'affichage et vous pourrez avec ces données réaliser de petites animations sur votre écran, écrites directement en code 6803.

• ***La résolution 64 × 64 points :*** dans ce cas la mémoire d'écran devra avoir une taille de 1024 octets. Chaque paire de deux bits correspondra à un point. Ainsi chaque octet décrira-t-il quatre points placés de la façon suivante :

P_3	P_2	P_1	P_0

Chaque point ayant sa couleur décrite sur deux bits, il est ainsi possible de représenter quatre couleurs différentes simultanément sur l'écran. Le format du mot transmis sur bus de données sera :

7	6	5	4	3	2	1	0
C_1	C_0	C_1	C_0	C_1	C_0	C_1	C_0
couleur de P_3		couleur de P_2		couleur de P_1		couleur de P_0	

Le bit 6 de BFFF permettra, quant à lui de sélectionner l'une ou l'autre de deux palettes de quatre couleurs. Pour accèder à ce mode 64 × 64 points il faut mettre le bit 5 de BFFF à "1", comme d'ailleurs pour tout mode haute résolution, et les bits 2, 3 et 4 à "0".

C_1	C_0	Css à 0	Css à 1
0	0	VERT	IVOIRE
0	1	JAUNE	BLEU CIEL
1	0	BLEU ROI	MAGENTA
1	1	ROUGE	ORANGE
		cadre vert	*cadre ivoire*

Il s'agit donc d'une résolution graphique 64 × 64, sur huit couleurs dont quatre sont utilisables simultanément, avec possiblilité de changement du jeu de couleurs.

• ***La résolution 128 × 64 points :*** il s'agit ici d'un mode graphique permettant l'utilisation simultanée de 2 couleurs seulement. En effet, chaque bit envoyé sur le bus de données décrira un point de l'écran. Or chaque bit ne pouvant prendre que deux valeurs, chaque point ne pourra avoir que deux couleurs possibles. Le format du code à envoyer sur le bus de données sera donc :

7	6	5	4	3	2	1	0
L_7	L_6	L_5	L_4	L_3	L_2	L_1	L_0

En fait, chaque bit Li indiquera pour le point correspondant si ce point est allumé (c'est-à-dire de la couleur du fond si le bit Li est à 1), ou si ce point est éteint (c'est-à-dire noir si le bit Li est à 0). C'est toujours le bit 6 de BFFF, connecté à la broche CSS du 6847 qui permet de choisir la couleur du fond : VERT si ce bit est à 0, IVOIRE sinon. Cette résolution graphique ne nécessite que 1024 octets pour fonctionner. On pourra donc utiliser également dans ce cas le programme donné plus loin pour faire des écrans de pavés graphiques, sans le modifier.

• ***La résolution 128 × 64 points en huit couleurs (dont quatre simultanées) :*** ce mode graphique nécessitera une mémoire d'écran de 2 k octets et permettra d'afficher 128 pixels par ligne d'affichage. Chacun des points sera défini de façon identique au mode 64 × 64 points. Pour atteindre ce mode haute résolution, il sera nécessaire de mettre les bits 3 et 5 de BFFF à "1" et les bits 2 et 4 à "0".

• ***Résolution 128 × 96 points en huit couleurs (dont quatre simultanément) :*** identique, dans son fonctionnement, à la résolution 64 × 64 points, ce mode d'affichage réclame une taille de la mémoire écran de 3 k octets. On atteint ce mode graphique en plaçant la valeur "1" dans les bits 2 et 5 de BFFF et "0" dans les bits 3 et 4 de cette même adresse, le jeu de couleurs étant sélectionné par le bit 6 (CSS).

• ***Résolution 128 × 192 points en deux couleurs :*** semblable à la résolution 128 × 64 en deux couleurs, en plaçant "1" dans les bits 2, 4 et 5 de BFFF et "0" dans le bit 3.

• ***Résolution 128 × 192 points en quatre couleurs simultanées :*** le principe reste toujours identique à celui de l'affichage en 64 × 64 points mais avec une taille de la mémoire d'écran atteignant 6 k octets ! L'extension mémoire de TANDY s'avère alors indispensalbe, mais quel plaisir de voir alors apparaître sur votre écran un dessin dans une haute résolution graphique véritable. Pour apprécier la finesse du dessin, on pourra déjà remplacer le contenu de l'adresse 36827 décimal (soit 8FDB hexa) du programme de génération d'écran donné plus loin, par la valeur 88 décimal, avant de l'exécuter, en ayant pris soin de placer aux bits 2, 3 et 5 du registre de contrôle du 6847 (placé à l'adresse BFFF hexa) la valeur "1", et de placer "00" dans le bit 4 de ce même registre de contrôle. Mais bien sûr, l'idéal est encore de réaliser vous-même, en utilisant la même technique et en le programmant en langage machine, votre propre dessin en mode haute résolution.

• ***Résolution 256 × 192 points en deux couleurs :*** c'est la plus haute résolution graphique à laquelle on puisse accèder sur ALICE ou MC10. C'est d'ailleurs également la plus haute résolution graphique que puisse générer le circuit MC 6847. Il est alors nécessaire de dis-

poser d'une mémoire d'écran de 6 k. Les dessins ainsi réalisés ne pourront être qu'en deux couleurs seulement. Pour sélectionner ce mode graphique, il faudra placer la valeur "1" dans les bits 2 et 5 du registre de contrôle du 6847 (adresse BFFF hexa).

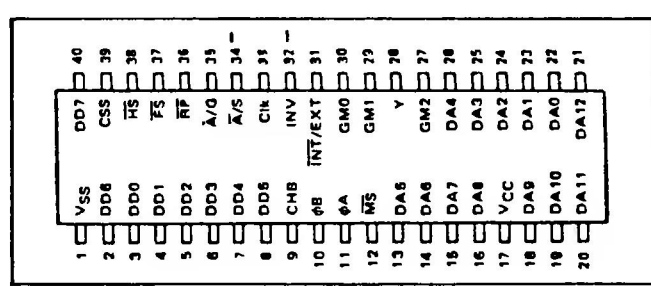
• ***Résolution 256 × 16 éléments en deux couleurs :*** il ne s'agit pas à proprement parler d'une résolution graphique, encore que les symboles obtenus puissent, peut-être, être utilisés pour réaliser des effets graphiques. En fait, il s'agit de la résolution semi-graphique à 6 éléments décrite précédemment. Or la réalisation technique d'ALICE et du MC10, qui connecte la broche GM2 et la broche INT/EXT du 6847 sur le même fil arrivant au bit 2 du registre de contrôle, nous interdit l'accès à un générateur de caractère externe au 6847 (cela nous aurait permis la redéfinition de caractères). En effet le rôle de la broche INT/EXT du 6847 est bien de choisir entre le générateur de caractère interne et un générateur de caractères externe. Or ici, on perd cette possibilité, mais cette broche étant couplée avec la broche GM2 (Graphic Mode N° 2) nous permet, en place de caractères redéfinis, l'accès à des bâtons très fins (256 bâtons par ligne) à condition que le bit 7 du bus de données soit maintenu à "0". On a vu en effet que dès qu'il était mis à "1", on passait en mode semi-graphique à 6 éléments. On aura ainsi sur le bus de données :

7	6	5	4	3	2	1	0
0	L6	L5	L4	L3	L2	L1	L0

Chaque Li correspondra à un bâtonnet. Si le Li correspondant est à "00", le bâtonnet sera noir (ou rouge si CSS est à "1") et si le Li correspondant est à "1" le bâtonnet sera vert (ou orange si CSS est à "1").

En conclusion de ce chapitre, on voit qu'ALICE, tout comme le TANDY MC10, possèdent plusieurs modes haute résolution et deux modes basse résolution qui en font des machines offrant de très belles possibilités graphiques. On peut cependant regretter un certain nombre de lacunes (peut-être voulues pour limiter la puissance de ce qui devrait rester, dans l'esprit des fabricants, un matériel de bas de gamme) telles que le fait d'avoir placé la mémoire écran à un endroit où on ne peut pas augmenter sa taille sans recouvrir la zone RAM nécessaire au Basic, ou encore d'avoir jumelé des broches du 6847 ensemble ce qui nous empêche d'exploiter au maximum les possibilités de ce circuit, alors que dans le registre de contrôle vidéo, deux bits demeurent inutilisés. On retrouve d'ailleurs ce problème pour la génération son (bit 7) qui nous ramène obligatoirement en basse résolution.

BROCHAGE DU 6847



REGISTRE CONTRÔLE 6847 — BFFF hexa								COULEURS			ÉCRAN		BUS DE DONNÉES	TAILLE MÉMOIRE ÉCRAN
SDN	Css	A/G	GM0	GM1	GM2 I/E	X	X	CARACTÈRES	FOND	CADRE	MODE D'AFFICHAGE	FORMAT		
X	0	0	X	X	0	X	X	vert* noir orange* rouge	noir* vert rouge* orange	noir	alphanumérique 32 × 16 caractères	standard 5 × 7 pixels	code ASCII 0 | | | | | | | connecté à INV (inverse si 0)	512 octets
	1													
X	0	0	X	X	1	X	X	L1 0 1	noir vert	noir	bâtonnets 256 × 16	0 L6 L5 L4 L3 L2 L1 L0	0 L6 L5 L4 L3 L2 L1 L0	512 octets
	1							0 1	rouge orange					
X	X	0	X	X	0	X	X	L1 C2 C1 C0 0 X X X 1 0 0 0 1 0 0 1 1 0 1 0 1 0 1 1 1 1 0 0 1 1 0 1 1 1 1 0 1 1 1 1	noir vert jaune bleu roi rouge ivoire bleu ciel magenta orange	noir	semi-graphique 64 × 32 éléments	L3 L2 L1 L0	couleur 1 C2 C1 C0 L3 L2 L1 L0 luminance des éléments 1 → allumé 0 → éteint	512 octets
X	0	0	X	X	1	X	X	L1 C 0 X 1 0 1 1	noir bleu roi rouge		semi-graphique 64 × 48 éléments	L5 L4 L3 L2 L1 L0	1 c L5 L4 L3 L2 L1 L0 1 couleur	512 octets
	1							0 X 1 0 1 1	noir mauve orange					
X	0	1	0	0	0	X	X	C1 C0 0 0 0 1 1 0 1 1	vert jaune bleu roi rouge	vert	graphique mode 1 64 × 64 éléments 4 couleurs	P3 P2 P1 P0	C1 C0 C1 C0 C1 C0 C1 C0 P3 P2 P1 P0	1 k octets
	1							0 0 0 1 1 0 1 1	ivoire bleu ciel magenta orange	ivoire				
X	0	1	1	0	0	X	X	L1 0 1	noir vert	vert	graphique mode 1 128 × 64 éléments mono	L7 L6 L5 L4 L3 L2 L1 L0	L7 L6 L5 L4 L3 L2 L1 L0	1 k octets
	1							0 1	noir ivoire	ivoire				
X	0	1	0	1	0	X	X	mêmes couleurs que graphique mode 1 en 4 couleurs		vert	graphique mode 2 128 × 64 éléments 4 couleurs	P3 P2 P1 P0	C1 C0 C1 C0 C1 C0 C1 C0	2 k octets
	1									ivoire				
X	0	1	1	1	0	X	X	mêmes couleurs que graphique mode 1 mono		vert	graphique mode 2 128 × 96 éléments mono	L7 L6 L5 L4 L3 L2 L1 L0	L7 L6 L5 L4 L3 L2 L1 L0	1 5 k octets
	1									ivoire				
X	0	1	0	0	1	X	X	mêmes couleurs que graphique mode 1 en 4 couleurs		vert	graphique mode 3 128 × 96 éléments 4 couleurs	P3 P2 P1 P0	C1 C0 C1 C0 C1 C0 C1 C0	3 k octets
	1									ivoire				
X	0	1	1	0	1	X	X	mêmes couleurs que graphique mode 1 mono		vert	graphique mode 3 128 × 192 éléments mono	L7 L6 L5 L4 L3 L2 L1 L0	L7 L6 L5 L4 L3 L2 L1 L0	3 k octets
	1									ivoire				
X	0	1	0	1	1	X	X	mêmes couleurs que graphique mode 1 en 4 couleurs		vert	graphique mode 4 128 × 192 éléments 4 couleurs	P3 P2 P1 P0	C1 C0 C1 C0 C1 C0 C1 C0	6 k octets
	1									ivoire				
X	0	1	1	1	1	X	X	mêmes couleurs que graphique mode 1 mono		vert	graphique mode 4 256 × 192 éléments mono	L7 L6 L5 L4 L3 L2 L1 L0	L7 L6 L5 L4 L3 L2 L1 L0	6 k octets
	1									ivoire				

L'AFFICHAGE SEMI-GRAPHIQUE SUR ALICE

MODE SEMI-GRAPHIQUE A QUATRE ÉLÉMENTS

Code décimal	Code hexa	Dessin	Code décimal	Code hexa	Dessin
128 - 144 160 - 176 192 - 208 224 - 240	80 - 90 A0 - B0 C0 - D0 E0 - F0		136 - 152 168 - 184 200 - 246 232 - 252	88 - 98 A8 - B8 C8 - D8 E8 - F8	
129 - 145 161 - 177 193 - 209 225 - 241	81 - 91 A1 - B1 C1 - D1 E1 - F1		137 - 153 169 - 185 201 - 247 233 - 253	89 - 99 A9 - B9 C9 - D9 E9 - F9	
130 - 146 162 - 178 194 - 210 226 - 242	82 - 92 A2 - B2 C2 - D2 E2 - F2		138 - 154 170 - 186 202 - 248 234 - 254	8A - 9A AA - BA CA - DA EA - FA	
131 - 147 163 - 179 195 - 241 227 - 247	83 - 93 A3 - B3 C3 - D3 E3 - F3		139 - 155 171 - 187 203 - 249 235 - 255	8B - 9B AB - BB CB - DB EB - FB	
132 - 148 164 - 180 196 - 242 228 - 248	84 - 94 A4 - B4 C4 - D4 E4 - F4		140 - 156 172 - 188 204 - 220 236 - 252	8C - 9C AC - BC CC - DC EC - FC	
133 - 149 165 - 181 197 - 243 229 - 249	85 - 95 A5 - B5 C5 - D5 E5 - F5		141 - 157 173 - 189 205 - 221 237 - 253	8D - 9D AD - BD CD - DD ED - FD	
134 - 150 166 - 182 198 - 244 230 - 250	86 - 96 A6 - B6 C6 - D6 E6 - F6		142 - 158 174 - 190 206 - 222 238 - 254	8E - 9E AE - BE CE - DE EE - FE	
135 - 151 167 - 183 199 - 245 231 - 251	87 - 97 A7 - B7 C7 - D7 E7 - F7		143 - 159 175 - 191 207 - 223 239 - 255	8F - 9F AF - BF CF - DF EF - FF	

MODE SEMI-GRAPHIQUE A QUATRE ÉLÉMENTS

Les éléments "éteints" du dessin sont représentés en noir ; quant aux éléments "allumés", leur couleur varie selon le code utilisé : dans l'ordre donné dans les colonnes "Codes", les valeurs correspondent respectivement aux couleurs suivantes :

ORDRE	COULEUR
	VERT
	JAUNE
	BLEU ROI
	ROUGE
	BLEU CIEL
	MAGENTA
	ORANGE

Il est évident que lorsqu'on utilise l'imprimante TP10 de TANDY, qui n'imprime qu'en noir et blanc, l'un quelconque des huit codes produira le dessin correspondant avec les éléments "allumés" en blanc sur le papier !

MODE SEMI-GRAPHIQUE A SIX ÉLÉMENTS

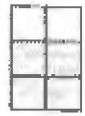
Les éléments "allumés" de chaque pavé graphique sont représentés en blanc sur le dessin. La couleur de ces éléments peut être, selon les valeurs indiquées dans les colonnes "Codes" :

Valeur du code	Css = 0	Css = 1
première	BLEU ROI	MAGENTA
deuxième	ROUGE	ORANGE

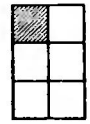
La valeur de Css, qui permet de sélectionner l'un ou l'autre des jeux de couleurs, est la valeur du bit 5 du mot situé à l'adresse BFFF hexa (49151 décimal). On choisira le contenu de cette adresse en utilisant l'instruction POKE du Basic.

Par exemple : pour mettre l'écran en basse résolution 64 × 64 points, deuxième jeu de couleurs, il faut mettre les bits 2 et 5 du mot situé à l'adresse 49151 décimal, à la valeur 1, soit : POKE 49151,68

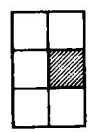
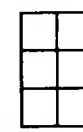
MODE SEMI-GRAPHIQUE A SIX ÉLÉMENTS

Code décimal	Code hexa	Dessin	Code décimal	Code hexa	Dessin
128 - 192	80 - C0		135 - 199	87 - C7	
129 - 193	81 - C1		136 - 200	88 - C8	
130 - 194	82 - C2		137 - 201	89 - C9	
131 - 195	83 - C3		138 - 202	8A - CA	
132 - 196	84 - C4		139 - 203	8B - 8C	
133 - 197	85 - C5		140 - 204	8C - CC	
134 - 198	86 - C6		141 - 205	8D - CD	

Code décimal	Code hexa	Dessin	Code décimal	Code hexa	Dessin
142 - 206	8E - CE		149 - 213	95 - D5	
143 - 207	8F - CF		150 - 214	96 - D6	
144 - 208	90 - D0		151 - 215	97 - D7	
145 - 209	91 - D1		152 - 216	98 - D8	
146 - 210	92 - D2		153 - 217	99 - D9	
147 - 211	93 - D3		154 - 218	9A - DA	
148 - 212	94 - D4		155 - 219	9B - DB	

Code décimal	Code hexa	Dessin	Code décimal	Code hexa	Dessin
156 - 220	9C - DC		163 - 227	A3 - E3	
157 - 221	9D - DD		164 - 228	A4 - E4	
158 - 222	9E - DE		165 - 229	A5 - E5	
159 - 223	9F - DF		166 - 230	A6 - E6	
160 - 224	A0 - E0		167 - 231	A7 - E7	
161 - 225	A1 - E1		168 - 232	A8 - E8	
162 - 226	A2 - E2		169 - 233	A9 - E9	

Code décimal	Code hexa	Dessin	Code décimal	Code hexa	Dessin
170 - 234	AA - EA		177 - 241	B1 - F1	
171 - 235	AB - EB		178 - 242	B2 - F2	
172 - 236	AC - EC		179 - 243	B3 - F3	
173 - 237	AD - ED		180 - 244	B4 - F4	
174 - 238	AE - EE		181 - 245	B5 - F5	
175 - 239	AF - EF		182 - 246	B6 - F6	
176 - 240	B0 - F0		183 - 247	B7 - F7	

Code décimal	Code hexa	Dessin	Code décimal	Code hexa	Dessin
184 - 248	B8 - F8		188 - 252	BC - FC	
185 - 249	B9 - F9		189 - 253	BD - FD	
186 - 250	BA - FA		190 - 254	BE - FE	
187 - 251	BB - FB		191 - 255	BF - FF	

EXTENSIONS DE BASIC

A l'étude de la ROM d'ALICE, il est apparu que plusieurs routines du système correspondaient à des instructions de Basic, accessibles directement avec la version de base, bien que non mentionnées dans le manuel du constructeur. Ces extensions peuvent être très utiles, aux amateurs de langage machine en particulier, car plusieurs d'entre-elles permettent d'écrire et de faire exécuter des routines en langage machine sur ALICE ou MC10, directement à partir de Basic. Nous nous intéresserons, dans ce chapitre à deux fonctions USR et VARPTR ainsi qu'à une instruction de branchement vers des routines écrites en langage machine : EXEC (qui est équivalente à l'instruction CALL qui existe sur d'autres machines), et à une extension de CLOAD, permettant de charger du langage machine depuis une cassette.

EXEC adresse décimale

Cette instruction effectue un branchement depuis Basic vers la sous-routine en langage machine, devant obligatoirement se terminer par l'instruction RTS pour que le retour au Basic s'effectue dans de bonnes conditions.
La syntaxe pour l'utilisation de EXEC est la suivante :

EXEC adresse décimale
ou EXEC (adresse décimale)
ou EXEC identificateur de variable numérique
ou EXEC (identificateur de variable numérique)

Lors de l'appel de EXEC, le système évalue l'adresse vers laquelle doit s'effectuer le branchement et place cette adresse sur deux octets en 16927 et 16928 en décimal avant d'effectuer le branchement vers cette adresse. Ainsi, si on écrit l'instruction EXEC sans adresse, le branchement est effectué automatiquement vers l'adresse contenue par les

octets 16927 et 16928 décimal (soit 421F et 4220 hexa).
On peut par exemple, sachant que la routine CLS est placé à l'adresse FBD4 soit 64468 décimal, pour tester le fonctionnement de EXEC, écrire - EXEC 64468 - qui exécutera un CLS. Mais bien sûr, on peut surtout écrire ses propres routines en langage machine vers lesquelles on pourra se brancher depuis un programme Basic. Autre intérêt de cette instruction, elle permet un branchement calculé. En fait, on a vu dans la syntaxe, que l'adresse de branchement pouvait être stockée dans une variable numérique. Cette adresse est traduite en entier et doit être comprise entre 0 et 65535 pour pouvoir loger sur 2 octets. Mais il est possible de remplacer cette adresse par une expression arithmétique, si l'adresse de branchement dépend par exemple du résultat d'un calcul :

Par exemple : l'appel à CLS pourra être réalisé par

```
10 I = 32000
20 EXEC 2*I+468
```

De même une réinitialisation complète du système peut être effectuée par un branchement à l'instruction de restart localisée à partir de l'adresse 63278. Il suffit de faire exécuter - EXEC 63278 - qui aura le même effet qu'une pression sur le bouton RESET. Toutefois cette instruction, de même que l'utilisation du bouton RESET, ne vous permettra pas à tout coup de reprendre le contrôle, en particulier après un mauvais branchement en langage machine. Il sera peut-être nécessaire alors de déconnecter votre ordinateur avant de le remettre sous tension, avec évidemment pour conséquence la perte totale des programmes et des données qui étaient dans la mémoire.
Cette instruction EXEC n'est pas faite pour permettre des échanges de valeur entre la routine et le Basic. Si vous voulez transmettre des valeurs à la routine ou restituer des résultats au Basic vous devrez le faire par des POKE et des PEEK à des adresses bien définies, ou utiliser USR.

USR (n)

Cette fonction appelle une sous-routine écrite en langage machine et lui transmet la valeur de l'argument "n", qui peut être fictif. Cette routine peut délivrer un résultat.
Il faut tout d'abord entrer la sous-routine en mémoire à partir du Basic par des POKE successifs. Au préalable, il est bon d'avoir réservé une zone protégée en haut de la mémoire grâce au petit programme utilitaire donné plus loin à cet usage. On détermine la taille de la routine en nombre d'octets et on retire cette valeur de la plus haute adresse de la RAM (4FFF en version de base ou 8FFF avec l'extension de mémoire 16 k). Le résultat de cette opération est l'adresse du début de l'espace à protéger.

Exemple : si on veut placer en mémoire une routine dont la longueur est 14 octets et réserver 6 octets supplémentaires pour des calculs soit 20 octets :

Sur version de base	Sur version étendue
4FFF − 14 = 4FEB	8FFF − 14 = 8FEB

On réservera alors cette zone de 20 octets, inaccessibles par Basic excepté par POKE, PEEK, USR et EXEC, de la façon suivante :

```
10 POKE 16977,235          EB hexa = 235 décimal
20 POKE 162,235
30 POKE 158,235
40 EXEC 63340
50 END
```

Lorsque le programme sera exécuté, le système sera réinitialisé en ayant réservé vos 20 octets en partie haute de la mémoire de 4FEC à 4FFF sur la version de base, ou de 8FEC à 8FFF sur la version étendue. A partir de ce moment, vous pouvez placer dans cette zone la routine écrite en langage machine par des POKE successifs. Des programmes Basic pourront être chargés, exécutés, effacés, sans danger pour le contenu de la zone protégée. Attention, l'utilisation de USR, tout comme celle de POKE ou EXEC, demande d'être familiarisé avec la programmation en langage machine. En effet, la moindre erreur de manipulation peut entrainer la destruction d'informations vitales pour le bon fonctionnement d'ALICE (pile, programmes, variables, pointeurs...) Dans ce cas, une action sur le bouton RESET s'avérera insuffisante pour restaurer le système et il sera alors presque indispensable de couper l'alimentation avant de pouvoir réutiliser ALICE dans de bonnes conditions. Lorsque vous voudrez utiliser la routine, vous utiliserez une instruction contenant USR comme par exemple :

100 PRINT USR(N) ou 50 X=USR(N) ou même 100 Y=USR(N)+X

Les octets 4216 hexa et 4217 hexa (16918 et 16919 en décimal), contiennent initialement l'adresse de la routine de traitement de l'erreur "Écriture Incorrecte Fonction" (FC ERROR). Il faut donc, avant d'utiliser USR, ranger par deux POKE aux adresses 16918 et 16919, l'adresse de la routine en langage machine. Dans notre exemple, si la routine doit être implantée à l'adresse 36847 décimal, c'est-à-dire à l'adresse 8FEF en hexadécimal, il faudra ranger 8F en hexa, soit 143 en décimal à l'adresse 16918, et EF en hexa, soit 239 en décimal à l'adresse 16919. On devra donc exécuter, avant d'appeler USR, les deux instructions suivantes :

POKE 16918,143 : POKE 16919,239

L'ordinateur, disposant de l'adresse de début de la routine, pourra alors commencer à l'exécuter.
Si vous souhaitez transmettre à la routine en langage machine, l'argument de la fonction USR, vous devrez appeler une routine d'acquisition de l'argument, localisée en EF4F en hexa. La valeur de l'argument sera alors convertie en entier et placée dans le registre d'index IX, sur deux octets. Cet argument devra donc être compris entre 0 et 65535. Toutefois, lorsque la valeur de l'argument dépasse 32768 elle sera interprétée comme un nombre négatif, complément à 65535 : par exemple, 38500 sera interprété par ALICE comme $-(65535 - 38500) = -27035$. Bien sûr, le passage de valeurs, du Basic à la routine, et inversement de la routine au Basic, peut également se faire par des POKE préalables à l'appel, à un endroit connu de la routine, et des PEEK, après l'exécution, à l'endroit où la routine a fourni le résultat.
Mais, la routine peut également terminer son exécution en restituant directement un résultat entier signé sur 2 octets par l'appel d'une routine système placée en ECE3 en hexadécimal. Dans l'exemple Y=USR(N), c'est ce résultat qui sera affecté directement à Y. Toutefois, avant d'appler la routine ECE3, il est nécessaire de placer le résultat sur deux octets

dans le double registre [A•B]. Si aucune valeur n'est transmise en retour, c'est la valeur de N qui sera affectée à Y.

Exemple :
```
10 I=36847
20 READ X
30 POKE I,X
40 IF X<>0 THEN I=I+1:GOTO 20
50 POKE 16918,143
60 POKE 16919,239
70 DATA 189,239,79,8,255,143,236,252,143,236,189,236,227,57,0
80 N=32600.478
90 PRINT USR(N)
100 Y=2*USR(19)+5
110 PRINT Y
120 END
```

Commentaires sur cet exemple

lignes 10 à 40 : placent aux adresses 36847 à 36860 décimal (8FEF à 8FFC) les instructions de la routine en langage machine. Chaque octet est ainsi lu et placé en mémoire par un POKE.

lignes 50 et 60 : placent aux adresses 16918 et 16919 décimal, les deux octets constituant l'adresse de début de la routine (8F en hexa = 143 en décimal et EF en hexa = 239 en décimal)

ligne 70 : valeurs placées en mémoire, correspondant aux instructions en langage machine dont le listing en assembleur 6803 serait :

```
8FEF BD EF 4F   JSR   ,('EF4F)   • prélève l'argument de USR, le transforme en entier et le place dans IX
8FF2 08         INX              • incrémente IX
8FF3 FF 8F EC   STX   ,('8FEC)   • place le contenu de IX (2 octets) en 8FEC et 8FED
8FF6 FC 8F EC   LDAD  ,('8FEC)   • place le contenu de 8FEC et 8FED dans le double registre [A•B]
8FF9 BD EC E3   JSR   ,(ECE3)    • restitue le résultat, contenu du double registre [A•B], au Basic
8FFC 38         RTS              • retour au basic
```

Cette routine (sans autre intérêt que pédagogique !...), prélève donc l'argument de USR, lui ajoute 1, et restitue le résultat au Basic.

lignes 80 et 90 : appel de la routine avec un argument décimal. Cet argument sera transformé en entier, et le résultat affiché par l'instruction de la ligne 90 sera donc 32601.

lignes 100-110 : USR(19) restitue le résultat 20 au Basic, et la valeur affectée à Y sera donc 45.

Si vous avez parfaitement compris l'utilisation de la fonction USR et de l'instruction EXEC sur ALICE, vous pouvez donc vous lancer dans la programmation de routines en langage machine, que vous pourrez appeler depuis vos programmes Basic, améliorant ainsi les performances de votre micro-ordinateur, dans des proportions importantes. De plus, en utilisant les indications données dans le paragraphe concernant le générateur d'écran, vous pourrez également améliorer la résolution graphique d'ALICE et peut-être réaliser de superbes animations...

VARPTR (nom de variable)

Une autre fonction très utile, et ne figurant pas dans le manuel d'initiation au Basic, bien que disponible sur ALICE et TANDY MC10, est VARPTR.
Cette fonction fournit l'adresse à laquelle la variable est stockée, lorqu'il s'agit d'une variable numérique, et l'adresse du descripteur de la variable, lorque cette variable est alphanumérique.
Si la variable n'a pas encore reçu de valeur, une erreur de type "Écriture Incorrecte de Fonction" (FC ERROR) se produira.

Variable numérique simple

VARPTR (variable numérique simple) fournit l'adresse A du premier octet de la valeur codée de cette variable. Ce codage est effectué sur cinq octets tels que :

[A] = exposant
[A+1] = octet le plus significatif de la mantisse
[A+2] = OPS2 de la mantisse
[A+3] = OPS3 de la mantisse
[A+4] = OMS de la mantisse

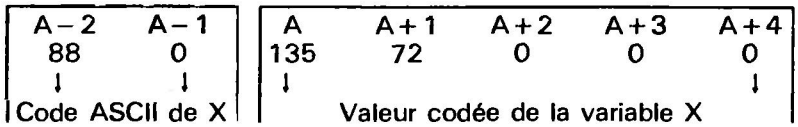
Vous trouverez plus de précision et quelques exemples de codage et de décodage des nombres en virgule flottante dans un autre chapitre de cet ouvrage. Ainsi, par exemple, si la variable X contient la valeur 100 (en décimal), l'instruction — PRINT VARPTR(X) — affichera l'adresse mémoire où est stockée la valeur codée de X, et en faisant exécuter la boucle suivante :

```
FOR I=VARPTR(X) TO VARPTR(X)+4:PRINT PEEK(I);:NEXT
```

vous verrez apparaître sur votre écran la ligne suivante :

135	72	0	0	0
↓	↓	↓	↓	
exposant	OPS1	OPS2	OPS3	OMS

Les deux octets précédant l'adresse A contenant l'exposant, et fournie par la fonction VARPTR, contiennent les codes ASCII des deux premiers caractères de l'identificateur de la variable.
Ainsi, pour l'exemple précédent, on aura :



Variable alphanumérique simple

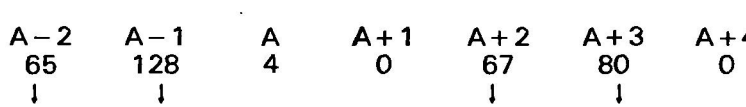
VARPTR (variable alphanumérique simple) fournit l'adresse A du premier octet du descripteur de cette variable dans la table des variables. Cette description est faite sur cinq octets (plus deux octets pour l'identificateur) sur le schéma suivant :

[A] = longueur en nombre de caractères de la chaine
[A+1] = zéro
[A+2] = octet le plus significatif de l'adresse à laquelle on trouve le premier caractère de la chaine
[A+3] = octet le moins significatif de l'adresse à laquelle on trouve le premier caractère de la chaine
[A+4] = zéro

Les octets [A−1] et [A−2] contiennent les codes ASCII des deux premiers caractères de l'identificateur de la variable alphanumérique.
En ce qui concerne [A−1], il contient en fait le code ASCII du deuxième caractère de l'identificateur, augmenté de 128, ce qui permettra, lors de l'exécution, de vérifier la compatibilité des types de variables dans une expression, par simple test sur la valeur du bit 7 du deuxième octet de l'identificateur : si ce bit 7 est à "0", il s'agira d'une variable numérique et si le bit 7 est à "1", il s'agira d'une variable alphanumérique.
Si l'on observe le descripteur de variable alphanumérique, on remarquera tout d'abord que la longueur de la chaine en nombre de caractères est codée sur un seul octet, ce qui limite la longueur des chaines alphanumériques à 255 caractères au maximum.
Quant à l'adresse à laquelle on trouve le premier caractère de la chaine (sur 2 octets), il peut s'agir soit d'une adresse dans la table source Basic lorsque la chaine est constante, par exemple — LET A$ = "TOTO" — ou d'une adresse dans la pile opérationnelle lorsque la chaine est variable, (bien entendu il s'agira alors de la zone "chaines" de la pile), par exemple — INPUT X$ — de telle sorte que le système minimise ainsi l'occupation mémoire. Ainsi, pour le programme suivant :

```
10 LET A$ = "TOTO"
20 INPUT X$
30 END
```

si après exécution du programme, on examine les descripteurs, on aura pour A$:



et pour X$:

88	128	10	0	143	245	0
code ASCII de X$	longueur			adr. dans pile op. 1^er caract. chaine		

Attention, dans le cas de X$, il s'agit d'un empilement, c'est-à-dire que si le premier caractère de la chaine est placé à l'adresse 8FF5 hexa, la pile commençant à l'adresse 8FFF, cela signifie qu'on a d'abord placé en fond de pile le dernier caractère de la valeur entrée, puis au-dessus, l'avant dernier caractère, etc... jusqu'au premier.
Pour visualiser les descripteurs de variables résultant de l'exécution du programme précédent, il suffisait après exécution du programme d'écrire la ligne suivante :

```
FOR I=VARPTR(A$)-2 TO VARPTR(X$)+4:PRINT PEEK(I);:NEXT I
```

Variable dimensionnée (ou tableau)

La fonction VARPTR, appliquée à une variable indicée, rend l'adresse du premier octet de l'élément de la variable indicée qu'elle indique. Ainsi, si on a défini une variable à une dimension L, et si on écrit — PRINT VARPTR(L(2)) — ALICE nous rendra l'adresse du deuxième élément de L. Il est par contre impossible de faire un VARPTR de l'ensemble d'une variable indicée.

Par exemple, si on a

```
10 DIM L(20)
...
100 PRINT VARPTR(L)
```

l'exécution de ce programme sera interrompue sur erreur à la ligne 100. En effet, VARPTR ne peut porter que sur des éléments de L, pris individuellement, du type : VARPTR(L(I)).
Pour plus de détails sur le codage des valeurs dimensionnées, on pourra se reporter au chapitre traitant du sujet, dans lequel des tableaux et des exemples permettront (je l'espère !...), d'éclaircir ce point particulier.

CLOADM

Nous terminerons l'examen des instructions au Basic avec un CLOADM, extension du CLOAD, qui permet de charger en mémoire centrale un programme en langage machine, et ceci directement à l'emplacement de votre choix, sans passer par le Basic et des octets en décimal. En fait, avec ce CLOADM, il est évidemment possible de charger n'importe quoi en mémoire : du langage machine, du texte (ce qui permet d'utiliser des fichiers-texte), des tableaux (voir la gestion de compte-chèque), etc... On doit cependant remarquer quelques particularités de fonctionnement de cette instructions si l'on veut pouvoir l'utiliser dans de bonnes conditions.
La syntaxe est la suivante : CLOADM "nom du fichier",adresse origine de l'implantation — nombre d'octets à transférer.
Ainsi, par exemple, si on a sauvegardé le contenu d'un écran sur cassette, on peut rechar-

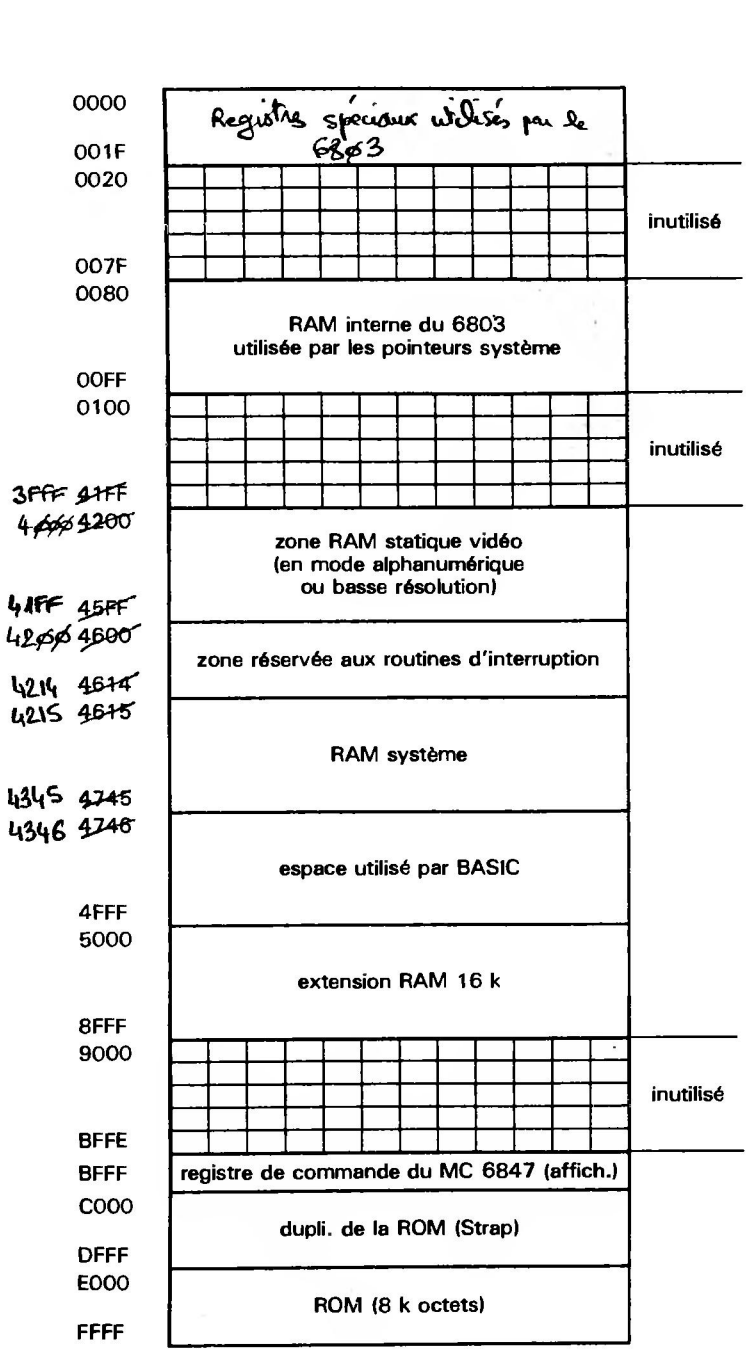
ger ce contenu sur l'écran par l'intermédiaire de l'instruction — CLOADM" ",16384−512 — 16384 étant l'adresse décimale du premier octet de l'écran en basse résolution, et 512 étant en décimal le nombre d'octets à transférer.
Après avoir vu la commande CLOADM qui vous permet de recharger en mémoire des octets que vous avez, au préalable, sauvegardés sur cassette, il n'est sans doute pas inutile d'évoquer justement la façon de sauvegarder le contenu d'une zone mémoire sur cassette. On pense aussitôt à une commande*CSAVEM qui aurait cette fonction. Oh surprise ! cette commande n'existe pas dans la ROM d'ALICE ! Ainsi donc, on peut charger une cassette mais on ne peut pas sauvegarder sur cassette une zone mémoire. C'est pour le moins surprenant et il n'est pas interdit de penser que cette lacune n'est en fait qu'une coupe qui a été faite dans la ROM pour une raison de place, afin de ne pas dépasser les 8 k octets de la ROM.
Afin de bénéficier de la souplesse de ces possibilités de sauvegarde sur cassette, je vous propose donc le petit programme suivant en langage machine qui vous permettra quand même d'effectuer cette sauvegarde.
L'exemple donné ci-dessous correspond à une sauvegarde d'écran :

```
CE  40  00    LDX   ,'4000
FF  42  6F    STX   ,('426F)
CE  42  00    LDX   ,'4200
FF  42  71    STX   ,'(4271)
C6  02        LDA   B,'02
7E  FC  4F    JMP   ,(FC4F)
```

Dans le registre X, on met d'abord l'adresse du premier octet à transférer, adresse qu'on stocke en 426F hexa (soit en 17007-17008 décimal), puis on stocke en 4271 hexa (soit en 17009-17010 décimal), l'adresse du dernier octet à transférer.
On place alors la valeur 02 dans le registre B avant de faire un branchement en FC4F, c'est-à-dire à l'intérieur de la routine CSAVE de la ROM. La valeur 02 qui avait été mise dans B sera alors placée à l'adresse 4267 hexa (soit 16999 décimal). C'est cette valeur (qui est en fait un indicateur), qui indiquera le type de bloc à transferer (basic, tableau ou code machine).
S'il s'agit d'un programme Basic, c'est la valeur 0 qui sera placée en 4267 hexa, et s'il s'agit d'un tableau numérique, c'est la valeur 04. Lors du chargement, c'est un test sur cette adresse qui permettra de détecter un "FC ERROR", c'est-à-dire une erreur sur le type de fichier.
Le programme ci-dessus, donné en assembleur, pourra être placé en mémoire par des POKE successifs et exécuté par la commande EXEC ou par l'intermédiaire d'un USR.
Je vous laisse imaginer les possiblilités qui s'ouvrent alors à vous, à partir d'un simple micro-ordinateur ALICE ou MC10 de base.

LA MÉMOIRE D'ALICE

Outre les 32 premiers octets, internes au 6803 (adresses '00 à '1F), qui contiennent les registres spéciaux permettant le contrôle et le bon fonctionnement du 6803, celui-ci contient également 128 octets de RAM interne, de l'adresse '80 à l'adresse 'FF dans lesquels le système place tous les pointeurs dont il a besoin pour permettre aux programmes de s'exécuter. Nous allons détailler ces différents pointeurs. La RAM externe qui se présente dans ALICE sous la forme de deux modules 4016 de 2 K octets chacun se trouve placé de l'adresse '4200 à l'adresse '4FFF et contient en particulier la zone mémoire écran au début de la RAM, c'est-à-dire de '4200 à '45FF, ce qui représente les 512 octets nécessaires à l'affichage en mode alphanumérique ou basse résolution graphique.
Lorsqu'on utilise l'extension mémoire de 16 K qui s'enfiche directement sur le connecteur d'extension, on obtient alors une RAM de 20 k octets (4 K de base + extension) se situant à l'adresse '4200 à l'adresse '8FFF. L'extension vient donc se placer immédiatement après la mémoire de base.
Quant à la ROM, il s'agit dans ALICE d'une PROM de type 2764, d'une capacité de 8 K octets, qui se place en haut de la zone adressable, c'est-à-dire de l'adresse 'E000 à l'adresse 'FFFF.
Nous allons étudier successivement dans ce chapitre chacune de ces zones mémoires en détaillant l'utilisation qui en est faite par le micro-ordinateur. Nous laisserons cependant de côté, dans cette étude, les registres spéciaux, adresses '0000 à '0020, dont nous avons déjà parlé longuement dans le premier chapitre, et que nous nous contenterons de lister ici. Revenons d'abord brièvement sur le plan général de la mémoire d'ALICE.



adresse hexa	adr. décimale
0000	0
001F	31
0020	32
007F	127
0080	128
00FF	255
0100	256
3FFF 41FF	16383
4000 4200	16384
41FF 45FF	16895
4200 4600	16896
4214 4614	16916
4215 4615	16917
4345 4745	17221
4346 4746	17222
4FFF	20479
5000	20480
8FFF	36863
9000	36864
BFFE	49150
BFFF	49151
C000	49152
DFFF	57343
E000	57344
FFFF	65535

LES REGISTRES SPÉCIAUX DU 6803

adresse hexa	adresse déci	registre
00	00	registre de direction Port 1
01	01	registre de direction Port 2
02	02	Port 1 d'E/S
03	03	Port 2 d'E/S
08	08	registre d'état temporisateur
09	09	OPS du compteur
0A	10	OMS du compteur
0B	11	OPS du registre sortie du temporisateur
0C	12	OMS sortie du temporisateur
0D	13	OPS entrée du temporisateur
0E	14	OMS entrée du temporisateur
10	16	registre format et débit communications série
11	17	registre d'état de l'interface série
12	18	registre de réception série
13	19	registre d'émission série
14	20	registre de contrôle RAM interne
15 à 1F	21 à 31	réservés

La zone comprise entre l'adresse '0020 et l'adresse '007F (soit 32 à 127 en décimal), est inutilisée sur ALICE. Il s'agit d'une zone RAM externe sur le 6803.
La zone RAM de 128 octets comprise entre les adresses '0080 et '00FF (soit 128 et 255 en décimal), est extrèmement intéressante car il s'agit de la zone dans laquelle le système, que nous détaillerons plus loin, place une grande partie des pointeurs dont il a besoin. En particulier, c'est dans cette zone qu'on trouvera tous les pointeurs Basic.

LES POINTEURS

On les trouve donc entre les adresses '80 et 'FF en hexadécimal, c'est-à-dire dans la zone comprise entre les adresses 128 et 255 en décimal.

Adresses		Nombre octets	Contenu
Hexa	décim		
82	130	1	Cet octet contient le nombre de caractères déjà saisis pendant la frappe d'une instruction BASIC. C'est sur cet octet que sera fait un test pour limiter la longueur des instructions (limitées à 128 caractères).
84	132	1	Cet octet sert à indiquer le type de variable en cours d'utilisation 0 pour une variable numérique ; 'FF pour une variable alphanumérique.
85	133	1	Lorsque plusieurs instructions figurent dans une même ligne de programmes, séparées par le signe :, on met la valeur '3A à cette adresse lors de l'exécution pour indiquer que l'instructions suivante n'est pas chaînée dans la TIP.
87	135	2	Zone de sauvegarde (pour registres).
89	137	2	Ce pointeur, sur deux octets, est un pointeur dans la pile pour l'instruction en cours.
8B	139	2	Contient l'adresse de la valeur de la variable active.
91	145	2	Pointeur sur le sommet de la pile opérationnelle.
93	147	2	Adresse du premier octet de la table des instructions de programme BASIC. Initialisée par le système à '4346 hexa.
95	149	2	Adresse du premier octet de la table des variables BASIC. Sert aussi de point de fin de table des instructions de programme + 1.
97	151	2	Contient l'adresse du troisième octet de la première variable indicée dans la table des variables : pointe donc sur le premier octet suivant le nom de la première variable indicée.
99	153	2	Adresse de fin de la table des variables : contient l'adresse du premier octet libre suivant la table des variables BASIC.
9B	155	2	Pointeur sur la base de la pile opérationnelle ; en fait contient l'adresse de l'octet situé immédiatement après la base de la pile opérationnelle.
9D	157	2	Pointeur sur le sommet de la zone chaînes de caractères de la pile, contient l'adresse du premier octet libre dans la zone chaîne.
9F	159	2	Pointeur sur le premier octet de la seconde chaîne empilée. Permet de restituer immédiatement la valeur de la chaîne en sommet de pile.
A1	161	2	Pointeur sur le haut de la mémoire vive, mis à jour lors de l'initialisation. Correspond à l'adresse de la base de la pile chaînes de caractères.

Adresses		Nombre octets	Contenu
Hexa	décim		
A3	163	2	Numéro de l'instruction sur laquelle le programme est arrêté. Mise à jour lorsque la touche BREAK est enfoncée ou par STOP ou END.
A5	165	2	Adresse du dernier octet de l'instruction exécutée. Contient également l'adresse de destination de la valeur lors d'une instruction POKE.
A7	167	2	Contient l'adresse du dernier octet de l'instruction précédant celle qui vient d'être arrêtée lors d'un BREAK. Pointe donc sur le séparateur.
A9	169	2	Adresse du premier octet de l'instruction en cours d'exécution dans la table des instructions de programme.
AB	171	2	Contient le numéro de l'instruction DATA en cours d'utilisation.
AD	173	2	Adresse du premier octet de la prochaine valeur à prélever dans le DATA en cours.
AF	175	2	Adresse du dernier octet entré dans le buffer d'entrée lors d'une entrée au clavier.
B1	173	2	Codes ASCII des deux premiers caractères du nom de la variable courante
B3	179	2	Contient l'adresse du premier octet de la variable utilisée. Dans le cas où il s'agit d'une variable alphanumérique, pointe sur le premier caractère de la chaîne.
B5	181	2	Contient l'adresse du premier octet suivant le nom de la variable dans la table des variables. S'il s'agit d'une variable numérique, pointe sur le premier octet de sa valeur et s'il s'agit d'une variable alphanumérique, pointe sur le premier octet (la longueur) de son descripteur dans la table des variables.
BB à C2	187 à 194	8	Zone de manœuvre dans laquelle on sauvegarde des pointeurs de début, de fin de chaînes lors de l'empilement d'une chaîne ou de la mise à jour de la table des instructions de programme (voir routines en ROM en 'E2C1 et 'E1FE).
C3 à C6	195 à 198	4	Zone de stockage intermédiaire d'adresses. Utilisé en particulier lors de la recherche d'une variable dans la table des variables.
C9 à CD	201 à 205	5	Zone de stockage de nombres en virgule flottante. Utilisé pour les passages de paramètres de fonction BASIC ainsi que pour la restitution des résultats de ces fonctions. Cette zone est également utilisée pour effectuer des conversions de type (ASCII en entier ou réciproquement).
CE à D1	206 à 209	4	Zone intermédiaire.
D2	210	2	Adresse du premier octet dans le buffer de sortie pour affichage.
DE	222	2	Adresse du dernier octet à prélever dans le buffer de sortie lors de l'affichage sur écran ou imprimante.
E2	226	2	Numéro de l'instruction Basic en cours d'exécution. Contient 'FFFF si l'instruction en cours d'exécution est en mode direct.

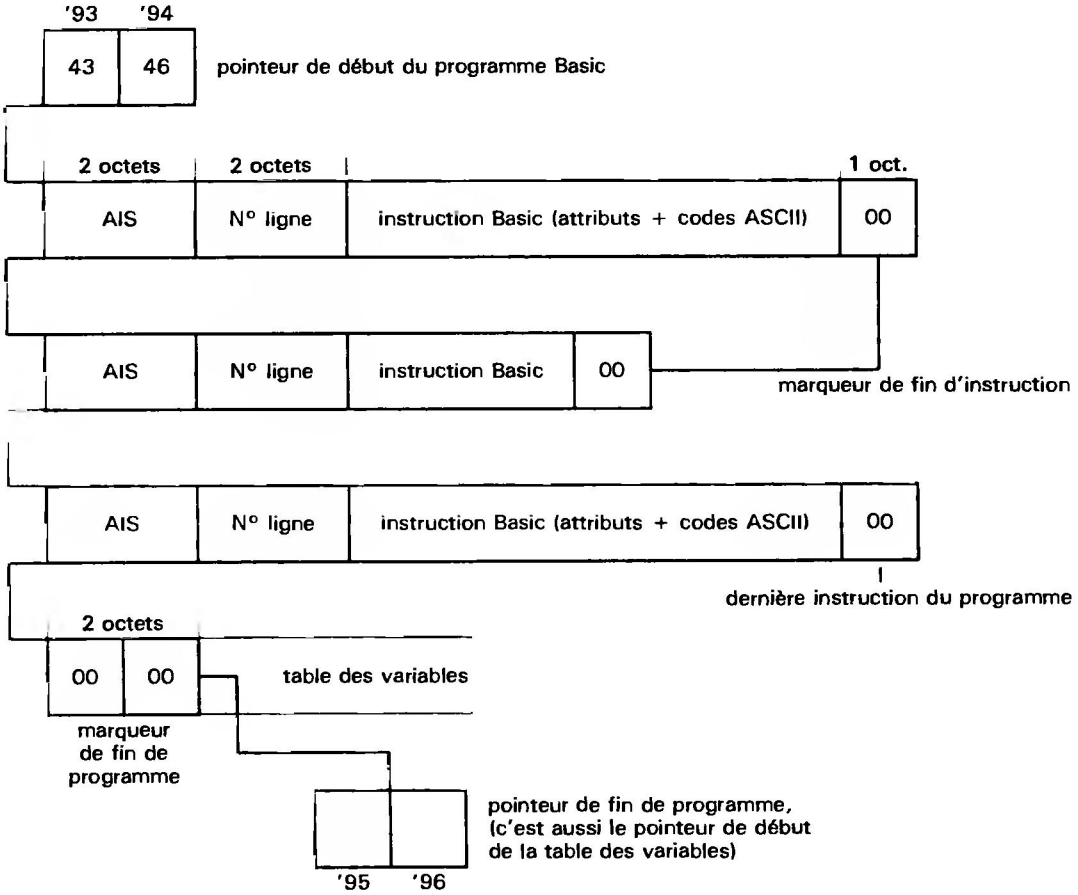
Adresses		Nombre octets	Contenu
Hexa	décim		
E4	228	2	Contient '1010.
E6	230	1	Numéro de ligne sur laquelle s'effectue l'affichage courant sur l'écran.
E7	231	1	Position courante sur la ligne d'affichage (varie de 0 à 31).
E8	232	1	Indicateur de dispositif de sortie. Si cet octet est à 0, la sortie s'effectue sur l'écran ; sinon il s'agit d'une sortie sur imprimante. En fait les routines en ROM, pour une sortie sur imprimante, placent la valeur 'FE dans cet indicateur.
EA	234	1	Mis à '55 lors de l'initialisation. Cette valeur est testée pour déterminer s'il s'agit d'un démarrage à chaud ou à froid. En cas de redémarrage à chaud, le programme se trouvant éventuellement en RAM n'est pas effacé.
EB à F8	235 à 248	14	Routine en langage machine permettant l'analyse, caractère par caractère des instructions entrées en mémoire. 00EB 7C 00 F5 INC ,('00F5) 00EE 26 03 BNE ,'00F3 00F0 7C 00 F4 INC ,('00F4) 00F3 B6 48 2E LDA A,('482E) 00F6 7E E1 C8 JMP ,('E1C8) La routine 'E1C8 vers laquelle s'effectue le branchement effectue un test sur le code prélevé.
'F4	244	1	Contient l'adresse de l'octet courant, en cours d'analyse, dans la routine précédente.
'F9 à 'FF	249 à 255	7	Ces sept octets sont inutilisés. Initialisés à zéro, ils peuvent être utilisés par vos propres routines comme zones de sauvegarde des pointeurs importants.

LA TABLE DES INSTRUCTIONS DE PROGRAMME BASIC

Le premier octet de cette table qui contient toutes les instructions du programme Basic se trouve, par adressage système à l'adresse '4346 (soit 17222 en décimal). L'origine de cette table est fixée par le système et l'adresse de cette origine est placée dans un pointeur sur deux octets situé à l'adresse : '93:'94 (soit en décimal 147 et 148). Le format des instructions de programme est le suivant :

- deux octets de chaînage contiennent l'adresse du premier octet de l'instruction suivante ;
- deux octets contiennent le numéro de l'instruction (ce qui permet dans un programme Basic d'écrire des numéros d'instructions compris entre 1 et 65535) ;
- puis on trouve l'instruction proprement dite dans laquelle chaque mot-clé est remplacé par son attribut hexadécimal (voir table des attributs). Par contre, si un nombre apparaît dans une instruction, il est représenté par la suite des codes ASCII des chiffres qui le composent ;
- à la fin de chaque instruction on trouve un octet à 0 qui sert de séparateur.

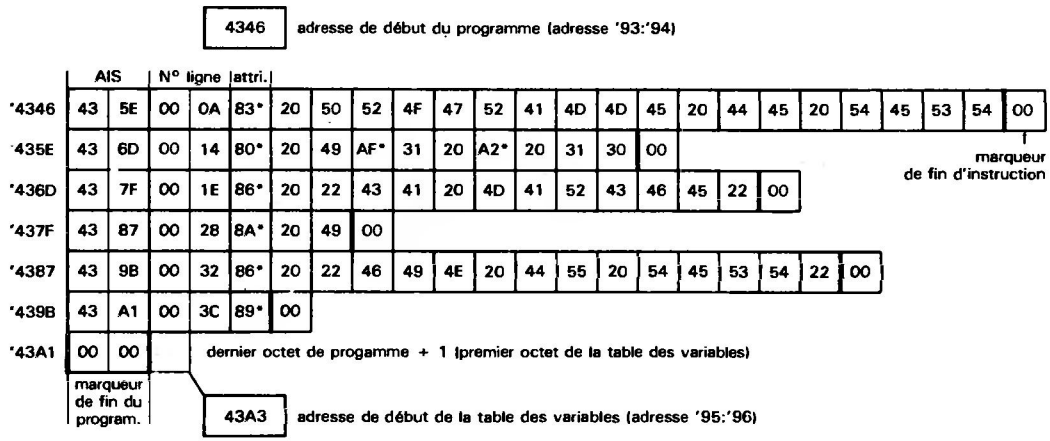
Il est à noter que lors de l'interprétation de la ligne entrée dans le buffer d'entrée, les blancs sont recopiés tels quels dans la table Basic, bien que n'ayant pas de signification propre, hormis ceux qui apparaissent entre guillemets.
La fin de la table des instructions est repérée par un marqueur de fin de fichier constitué par deux octets à 0. Cette adresse de fin du programme + 1 figure dans le pointeur situé en '95:'96 (soit en décimal aux adresses 149 et 150).
Le schéma suivant montre en détail la structure de cette table dans la RAM d'ALICE.



• ***AIS - adresse de l'Instruction Suivante.***
cette adresse est donnée sur deux octets sous le format OPS-OMS et est évaluée en faisant AIS = OPS * 256 + OMS.
• ***OPS - octet le plus significatif***
• ***OMS - octet le moins significatif***
Le numéro de ligne, sur deux octets, est également donné sous le même format que l'AIS.

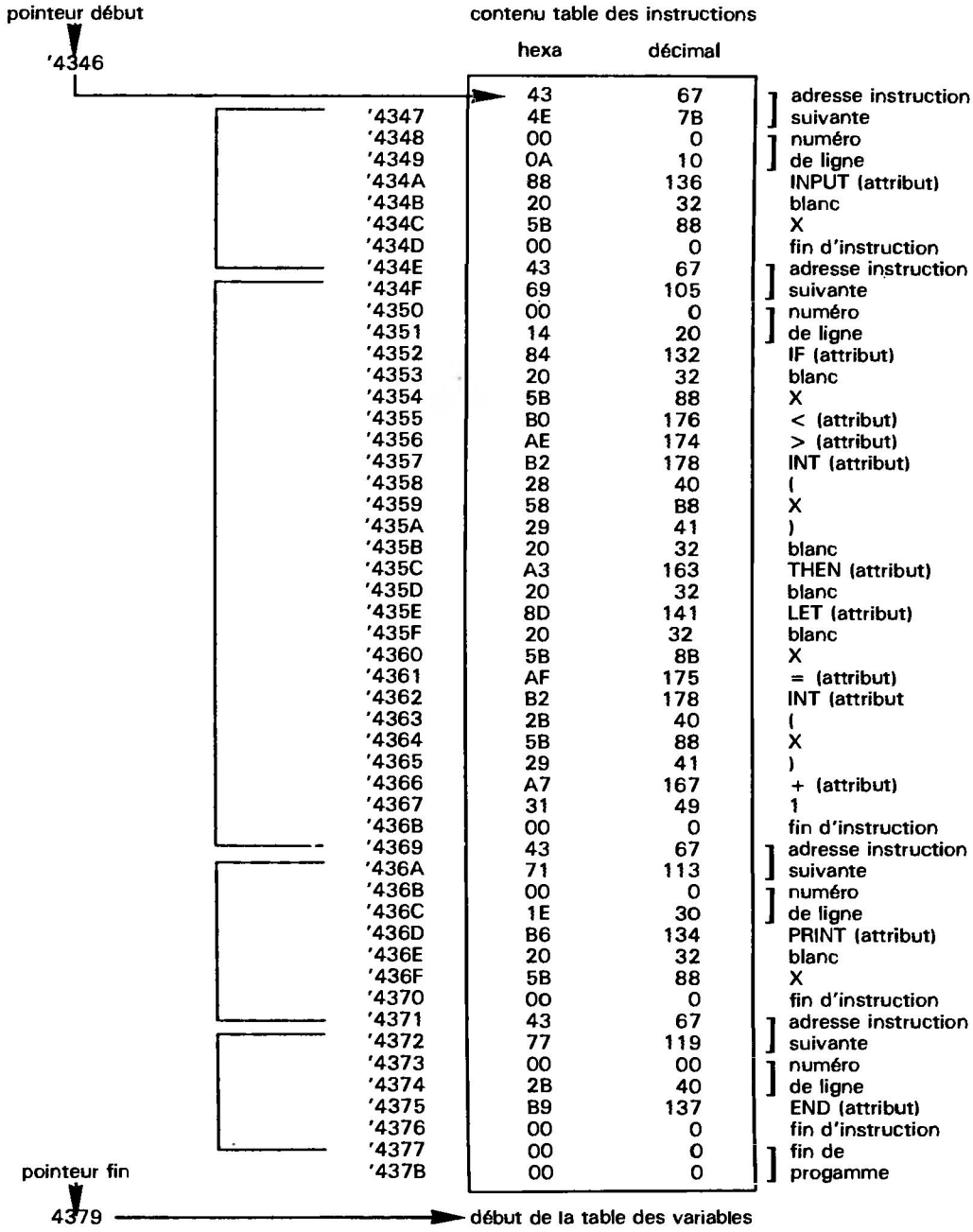
Nous allons maintenant donner des exemples de progammes Basic, en examinant dans ces cas précis les contenus de la table des instructions Basic ainsi que des pointeurs de début et de fin de progamme.

Exemple 1
```
10 REM PROGRAMME DE TEST
20 FOR I=1 TO 10
30 PRINT "CA MARCHE"
40 NEXT I
50 PRINT "FIN DU TEST"
60 END
```



Dans cet exemple, les attributs des mots-clés Basic apparaissent avec un astérisque (*). Les codes ASCII ainsi que les Adresses d'Instruction Suivante et les numéros de ligne sont donnés en hexadécimal.

Exemple 2
```
10 INPUT X
20 IF X<>INT(X) THEN LET X=INT(X)+1
30 PRINT X
40 END
```


pointeur début
'4346
contenu table des instructions
hexa
décimal
43 67 adresse instruction
'4347 4E 7B suivante
'4348 00 0 numéro
'4349 0A 10 de ligne
'434A 88 136 INPUT (attribut)
'434B 20 32 blanc
'434C 5B 88 X
'434D 00 0 fin d'instruction
'434E 43 67 adresse instruction
'434F 69 105 suivante
'4350 00 0 numéro
'4351 14 20 de ligne
'4352 84 132 IF (attribut)
'4353 20 32 blanc
'4354 5B 88 X
'4355 B0 176 < (attribut)
'4356 AE 174 > (attribut)
'4357 B2 178 INT (attribut)
'4358 28 40 (
'4359 58 B8 X
'435A 29 41)
'435B 20 32 blanc
'435C A3 163 THEN (attribut)
'435D 20 32 blanc
'435E 8D 141 LET (attribut)
'435F 20 32 blanc
'4360 5B 8B X
'4361 AF 175 = (attribut)
'4362 B2 178 INT (attribut
'4363 2B 40 (
'4364 5B 88 X
'4365 29 41)
'4366 A7 167 + (attribut)
'4367 31 49 1
'436B 00 0 fin d'instruction
'4369 43 67 adresse instruction
'436A 71 113 suivante
'436B 00 0 numéro
'436C 1E 30 de ligne
'436D B6 134 PRINT (attribut)
'436E 20 32 blanc
'436F 5B 88 X
'4370 00 0 fin d'instruction
'4371 43 67 adresse instruction
'4372 77 119 suivante
'4373 00 00 numéro
'4374 2B 40 de ligne
'4375 B9 137 END (attribut)
'4376 00 0 fin d'instruction
'4377 00 0 fin de
pointeur fin
'437B 00 0 progamme
4379
début de la table des variables

LA TABLE DES VARIABLES BASIC

Les variables Basic sont stockées en mémoire vive, immédiatement après la table des instructions de programme, à l'exception des variables alphanumériques qui sont stockées dans la zone "chaîne" de la pile ou, lorsqu'il s'agit de constantes, dans la table même des instructions de programme. On ne trouvera alors dans la table des variables qu'un descripteur de la variable.
Les variables sont placées dans la table des variables dans l'ordre de leur apparition dans le programme, en prenant soin toutefois de placer d'abord toutes les variables simples, puis ensuite les variables indicées. Pour améliorer la vitesse d'exécution on aura intérêt à placer en tête de la table, c'est-à-dire à utiliser en premier, les variables les plus utilisées dans le programme. En effet, à l'exécution, la recherche de la variable dans la table s'effectuant séquentiellement, le temps d'accès à une valeur située en début de table sera évidemment plus court que si cette dernière est placée en fin de table. On pourra ainsi, par exemple, initialiser en début de programme, les variables qui serviront le plus souvent.
Nous allons maintenant voir le format dans lequel sont stockées les variables dans la table, en commençant par les variables simples, puis ensuite les variables indicées.

LES VARIABLES SIMPLES

Le début de la table des variables est repéré par un pointeur sur deux octets, placé à l'adresse '95 et qui contient l'adresse du premier octet de cette table, c'est-à-dire, nous allons le voir, l'adresse du premier octet du premier identificateur, c'est-à-dire du nom de la première variable rencontrée dans le programme BASIC. Notons au passage que plusieurs pointeurs sont en rapport avec la table des variables et en particulier celui qui, placé à l'adresse '99, sur deux octets, pointe sur le dernier octet de la table des variables.
Parmi les variables simples, on distingue encore deux types : les variables numériques simples et les variables alphanumériques simples. Intéressons-nous d'abord aux variables numériques simples.

LES VARIABLES NUMÉRIQUES SIMPLES

Pour les variables, un seul format est utilisable en BASIC : le format virgule flottante. En effet le BASIC d'ALICE n'autorise pas le format entier (qui est pourtant largement utilisé par les routines de la ROM), ni les formats simple et double précision. Ce format unique, représentation d'une variable numérique, nécessite en tout sept octets dont deux servent à coder l'identificateur de la variable.
Le premier octet contiendra le code ASCII du premier caractère de l'identificateur et le deuxième octet contiendra le code ASCII du deuxième caractère de l'identificateur s'il existe et 0 sinon.
Le format de représentation de la valeur numérique de la variable comprend deux parties : l'exposant et la mantisse. L'exposant est représenté sur un octet et la mantisse sur quatre octets en commençant par les octets les plus significatifs, ce qui nous donne la représentation suivante :

DÉTERMINATION DU FORMAT DE STOCKAGE DE J = −0.875

Conversion en binaire	−.11100000
Représentation normalisée ($M*2^E$)	$-.11100000*2^0$
Exposant normalisé = exposant binaire + 128	128 (80 hexa)
Nombre négatif, donc laisser premier bit tel quel	−.11100000
	E0 hexa
Format de stockage	80 E0 00 00 00 hexa
	exposant OPS1 OPS2 OPS3 OPS4
	128 224 0 0 0 décim.

PROCÉDURE DE DÉCODAGE D'UN NOMBRE EN VIRGULE FLOTTANTE

Il peut être intéressant également de décoder un nombre stocké en mémoire vive sous la forme présentée ci-dessus. On procédera alors d'une façon exactement inverse de la précédente, c'est-à-dire :
- écriture de la mantisse en binaire ;
- si le premier chiffre est 0, alors le nombre est positif. Dans ce cas convertir le bit en 1, sinon le laisser tel quel (le nombre est alors négatif) ;
- détermination de l'exposant binaire = exposant normalise − 128 ;
- décalage du point binaire du nombre de positions correspondantes ;
- conversion du résultat en décimal.

Afin de bien comprendre le fonctionnement de cette procédure de décodage, nous allons étudier deux exemples simples :

DÉCODAGE EN DÉCIMAL DU NOMBRE STOCKÉ SOUS LA FORME 83 20 1E 00 00 hexa ou 131 32 30 0 0 décim.

Écriture de la mantisse en binaire	.001000000011110
Le nombre est positif (premier bit = 0) → conversion en 1	.101000000011110
Détermination de l'exposant : 131 − 128	+3
Décalage du point binaire de 3 positions à droite	101.000000011110
Conversion en décimal :	
$2^2 + 2^0 + 2^{-9} + 2^{-10} + 2^{-11} + 2^{-12}$	+5.00366211

N° octet	Contenu
1	Code ASCII premier caractère du nom de la variable.
2	Code ASCII second caractère du nom de la variable (ou zéro).
3	Exposant.
4	OPS de la mantisse (OPS1).
5	OPS suivant de la mantisse (OPS2).
6	OPS suivant de la mantisse (OPS3).
7	OMS de la mantisse.

OPS : Octet le Plus Significatif.
OMS : Octet le Moins Significatif.

PROCÉDURE DE CODAGE D'UN NOMBRE EN VIRGULE FLOTTANTE

Nous venons de voir qu'un nombre en virgule flottante est codé sous la forme mantisse et exposant. On peut donc dire que tout nombre en virgule flottante peut s'exprimer sous la forme :

$$N = M*2^E$$

où M représente la mantisse et E l'exposant.
Pour coder un nombre en virgule flottante, il va donc falloir déterminer M et E tels que $N = M*2^E$. On procédera donc de la façon suivante :
- conversion en binaire du nombre à coder ;
- représentation normalisée sous la forme $\pm 0.\text{mantisse} \times 2^{\text{exposant}}$;
- détermination de l'exposant normalisé = exposant binaire + 128 ;
- si le nombre est positif, mise à zéro du premier bit de la mantisse ; si le nombre est négatif, laisser ce premier bit tel quel ;
- mise au format définitif de la mantisse sur 4 octets.

Afin de mieux voir le fonctionnement de cette procédure de codage, nous allons étudier deux exemples :

DÉTERMINATION DU FORMAT DE STOCKAGE DE J = +5.75

1. Conversion en binaire	101.11
2. Représentation normalisée $M*2^E$	$+.10111*2^3$
3. Exposant normalisé = exposant binaire + 128	131 (83 hexa)
4. Nombre positif donc mise à 0 du premier bit	.00111000
	38 hexa
5. Format de stockage	83 38 00 00 00 hexa
	exposant OPS1 OPS2 OPS3 OMS
	131 56 0 0 0 décimal

DÉCODAGE EN DÉCIMAL DU NOMBRE STOCKÉ SOUS LA FORME
7F C0 00 00 00 hexa ou 127 192 0 0 0 décimal

Écriture de la mantisse en binaire	.11000000
Le premier bit est à 1 → le nombre est négatif	−.11000000
Détermination de l'exposant binaire 127 − 128	−1
Décalage du point binaire d'une position vers la gauche	−.011000000
Conversion en décimal : $2^{-2} + 2^{-3} = -.375$	−.375

Si l'on souhaite décoder un nombre stocké en virgule flottante par l'intermédiaire d'un programme BASIC, on devra procéder de la façon suivante :
- Calculer l'adresse de l'exposant en faisant VARPTR (nom de la variable). (Pour l'utilisation de VARPTR voir le chapitre sur les instructions cachées de BASIC).
- Par PEEK de cette adresse on obtient l'exposant normalisé. Pour obtenir l'exposant binaire, retirer 128.
- Calculer OPS1,... jusqu'à OMS par PEEK (adresse + 1) jusqu'à PEEK (adresse + 4).
- Convertir en binaire OPS1, OPS2, OPS3 et OMS.
- Calculer alors la position du point binaire en faisant un décalage à droite à partir du bit de poids fort de OPS1, d'autant de positions que la valeur de l'exposant binaire.
- Transformer le bit de poids fort de OPS1 en 1 s'il était nul (le nombre est alors positif) et le laisser tel quel si la valeur était 1 (le nombre est alors négatif).
- Conversion du nombre ainsi obtenu en décimal pour calcul des puissances de 2 correspondantes.

LES VARIABLES ALPHANUMÉRIQUES SIMPLES

En fait, pour les variables alphanumériques simples, il ne s'agit pas, dans la table des variables, de stocker la valeur de ces variables, mais seulement un descripteur pour chacune d'entre elles. Pour des raisons de simplification sans doute, le descripteur d'une variable alphanumérique simple nécessite sept octets, tout comme une variable numérique. Sur ces sept octets, seuls cinq d'entre eux sont utilisés, les deux autres restant en principe à 0. Le premier octet contiendra le code ASCII du premier caractère de l'identificateur de la variable, tandis que le deuxième octet contiendra, s'il existe, le code ASCII du deuxième caractère de l'identificateur, augmenté de 128 (ce qui permettra immédiatement de détecter les variables alphanumériques dans la table des variables).
Le descripteur comprend ensuite deux parties : la longueur en nombre de caractères de la chaîne décrite et le pointeur sur le premier caractère de la chaîne (sur deux) octets. On peut au passage noter que ce pointeur pourra aussi bien adresser la pile OP que le clavier. En effet, si la chaîne est constante (si sa valeur est affectée par le programme : par exemple LET X$ = "TOTO"), la suite des caractères qui composent cette chaîne apparaît dans la table des instructions de programme ; il est alors inutile de recopier cette valeur dans la zone "chaînes de caractères" de la pile, et c'est donc l'adresse à laquelle est stocké

le premier octet de la valeur de la chaîne qui sera placée dans le descripteur. Si, par contre, la chaîne est variable, son contenu sera toujours placé dans la zone "chaînes de caractères de la pile", ce qui minimise l'espace mémoire occupé. Nous reparlerons plus en détail de la pile, un peu plus loin, mais il est bon de rappeler que par défaut la zone "chaînes de caractères" de la pile n'a une capacité que de 100 octets.
Le descripteur d'une variable alphanumérique aura donc le format suivant dans la table des variables :

N° octet	Contenu
1	Codes ASCII du premier caractère du nom de la variable.
2	Codes ASCII du deuxième caractère du nom de la variable (ou 0) + 128.
3	Longueur de la chaîne de caractères.
4	0
5	OPS de l'adresse du premier caractère de la chaîne.
6	OMS de l'adresse du premier caractère de la chaîne.
7	0

OPS = Octet le plus significatif
OMS = Octet le moins significatif

Notons que la zone "chaîne de caractères" de la pile reçoit, en fond de pile, le dernier caractère de la chaîne et en sommet le premier caractère. En d'autres termes, pour obtenir le contenu d'une variable chaîne de caractères, on recherche l'adresse de son descripteur par VARPTR (nom de la variable), ensuite PEEK (adresse) nous fournira la longueur de cette chaîne et PEEK (Adresse + 2) * 256 + PEEK(Adresse + 3) nous donnera l'adresse du premier caractère de la chaîne. Pour accéder à la chaîne entière, on pourra exécuter la ligne suivante :

```
FOR I= 256*PEEK(ADR+2)+PEEK(ADR+3) TO 256*PEEK(ADR+Z)+PEEK(ÀDR+3)
+PEEK(ADR): PRINT CHR$(I);: NEXT I
```

où ADR est mis pour VARPTR (nom de la variable).

LES VARIABLES INDICÉES

Le début de la table des variables indicées est repéré par un pointeur (sur deux octets) situé à l'adresse '97 (ou 151 en décimal). Ce pointeur contient l'adresse du 3e octet de la table des variables indicées, c'est-à-dire qu'il pointe sur l'octet suivant le nom de la première variable indicée. Seront placées dans cette table des variables indicées, toutes les variables ayant reçu un dimensionnement explicite (par DIM) ou implicite. La table conserve les noms des variables sur deux octets, le nombre total d'octets utilisés par la variable ainsi que la valeur de chacun de ses éléments. Nous allons de nouveau distinguer deux cas selon qu'il s'agit d'une variable numérique ou d'une variable alphanumérique.

LES VARIABLES NUMÉRIQUES

Là encore, chaque élément du tableau sera stocké en format virgule flottante comme s'il s'agissait d'une variable simple. Examinons la structure d'un tableau numérique :

Les deux premiers octets contiendront les codes ASCII des deux premiers caractères du nom de la variable. Les deux octets suivants contiennent la longueur totale en nombre d'octets du tableau. L'octet suivant indique le nombre de dimensions du tableau et ensuite on trouve autant de fois deux octets qu'il y a de dimensions, chacun de ces blocs indiquant le nombre d'éléments dans chaque dimension en commençant par la dernière. Notons que lorsqu'on utilise un tableau, l'élément de rang 0 est toujours présent dans le tableau, ce qui peut être, si on ne l'utilise pas, un facteur de perte de place mémoire. Enfin on trouve, codés sur cinq octets, comme des variables simples, chaque élément du tableau. On peut donc résumer la structure de la table pour une variable indicée de la façon suivante :

N° octet	Contenu
1	Code ASCII du premier caractère du nom de la variable.
2	Code ASCII du deuxième caractère du nom de la variable (ou zéro).
3	OPS de la longueur du tableau.
4	OMS de la longueur du tableau. Longueur du tableau = 256*OPS + OMS représente le nombre d'octets pour passer à la variable indicée suivante.
5	Nombre de dimensions du tableau (o<n<255).
6 à n	Valeur maximale du nombre d'éléments de chaque dimension + 1, en commençant par la dernière : sur deux octets sous la forme OPS OMS.
n+1 à p	Valeur de chaque élément du tableau en partant de la valeur minimale du premier index pour arriver à la valeur maximale du dernier index. Chaque valeur est représentée sur cinq octets.

Ce tableau signifie que si on a déclaré un tableau T(2,3), l'ordre dans lequel les éléments seront stockés en mémoire vive sera :
T(0,0) → T(1,0) → T(2,0) → T(0,1) → T(1,1) → T(2,1) → T(0,2) → T(1,2) → T(2,2) → T(0,3) → T(1,3) → T(2,3).
On peut calculer ainsi facilement le nombre d'octets nécessaires au stockage d'un tableau en mémoire vive déclarée par DIM T(n1, n2, ..., nk).
Nombre d'octets = $2+2+1+(2*k)+(2_1+1)(n_2+1) \ldots (n_k+1)*5$
d'où $N = 5+2k+5(n_1+1)(n_2+1) \ldots (n_k+1)$
En utilisant cette formule, on voit que le plus petit tableau possible de la forme DIM T(1) nécessitera $N = 5+2+5\times 2 = 17$ octets.
Le programme suivant montre un exemple de stockage de tableau à deux dimensions : il s'agit d'un tableau T dimensionné (2,3).

Lignes 20 à 80 — Remplissage du tableau ligne par ligne avec les 12 premiers entiers.

Lignes 90 à 125 — Impression des paramètres du tableau stockés en RAM :

```
84  0   → Code ASCII de T (nom du tableau)
 0 69   → Nombre total d'octets occupés par T en mémoire.
    2   → Nombre de dimensions du tableau.
 0  4   → Nombre d'éléments sur la deuxième dimension.
 0  3   → Nombre d'éléments sur la première dimension.
```

Lignes 130 à 190 Impression des éléments du tableau dans l'ordre où ils sont stockés en mémoire vive. Chacun d'entre eux étant formaté en virgule flottante sur cinq octets.

Lignes 1, 165 et 177 Servent à afficher la valeur de l'élément de tableau correspondant aux cinq octets en virgule flottante.

```
1 T$="T(0,0) ="
10 DIM T(2,3)
20 K=0
30 FOR I=0 TO 2
40 FOR J=0 TO 3
50 LET T(I,J)=K
60 LET K=K+1
70 NEXT J
80 NEXT I
90 FOR I=VARPTR(T(0,0))-9 TO VAR
PTR(T(0,0))-1
100 LPRINTPEEK(I);
110 NEXT I
120 LPRINT:LPRINT
125 LPRINT:LPRINT
126 K=0:L=0
130 FOR I=VARPTR(T(0,0)) TO VARP
TR(T(0,0))+60 STEP 5
140 FOR J=0 TO 4
150 LPRINTPEEK(I+J);
160 NEXT J
165 LPRINT TAB(20);T$;T(K,L)
175 K=K+1
176 IF K>2 THEN K=0:L=L+1
177 T$=LEFT$(T$,2)+RIGHT$(STR$(K
),1)+","+RIGHT$(STR$(L),1)+RIGHT
$(T$,3)
180 NEXT I
190 END
```

CONTENU DU TABLEAU T APRES 'RUN'

0	1	2	3
4	5	6	7
8	9	10	11

FORMAT DE STOCKAGE DE T EN RAM.

84	0	0	69	2	0	4	0	3

0	0	0	0	0
131	[illegible]	0	0	0
132	[illegible]	0	0	0
129	[illegible]	0	0	0
131	32	0	0	0
132	16	0	0	0
130	0	0	0	0
131	64	0	0	0
132	32	0	0	0
130	64	0	0	0
131	96	0	0	0
132	48	0	0	0

T(0,0) = 0
T(1,0) = 4
T(2,0) = 8
T(0,1) = 1
T(1,1) = 5
T(2,1) = 9
T(0,2) = 2
T(1,2) = 6
T(2,2) = 10
T(0,3) = 3
T(1,3) = 7
T(2,3) = 11

LES VARIABLES ALPHANUMÉRIQUES

Le format de stockage sera identique à celui des tableaux numériques avec la différence qu'ici chaque élément ne contient pas la valeur mais un descripteur de la chaîne comportant la longueur de celle-ci et l'adresse à laquelle on trouve le premier caractère de chaque élément.
La structure d'un tableau alphanumérique stocké en mémoire sera donc la suivante :

N° octet	Contenu
1	Code ASCII du premier caractère du nom de la variable.
2	Code ASCII du second caractère du nom de la variable (ou zéro) + 128.
3	OPS de la longueur du tableau.
4	OMS de la longueur totale du tableau (y compris le nom).
5	Nombre de dimensions du tableau.
6 à n	Valeur maximale de chaque dimension + 1 (pour tenir compte des éléments de rang 0) en commençant par la dernière à droite jusqu'à la première à gauche, sous la forme OPS.OMS.
n+1 à p	Descriptif de chaque élément du tableau sous la forme 1. Longueur de la chaîne sur un octet 2. 0 3. Adresse du 1er caractère de l'élément OPS 4. Adresse du 1er caractère de l'élément OMS 5. 0 } autant de fois que d'éléments

Les commentaires sur l'ordre dans lequel sont stockés les éléments ainsi que sur l'évaluation de la taille d'un tableau, fait précédemment pour un tableau numérique, demeurent valables dans le cas d'un tableau alphanumérique.
A titre d'exemple, on pourra se reporter au listing suivant qui correspond au format de stockage d'un tableau T$ déclaré (2,3) et dans lequel on a placé les 12 premières lettres de l'alphabet.
Dans l'en-tête on trouve successivement :

```
84 128   → Nom T$ du tableau.
 0  69   → Longueur totale du tableau en nombre d'octets.
       2  → Nombre de dimensions du tableau.
 0   4   → Nombre d'éléments sur la deuxième dimension du tableau.
 0   3   → Nombre d'éléments sur la première dimension du tableau.
```

Ensuite, pour chaque élément, on reconnaît :
- La longueur de l'élément (ici 1 car chaque élément contient un seul caractère qui est une lettre de l'alphabet).
- 1 octet à 0.
- L'adresse du premier caractère de l'élément (ici les éléments sont dans la zone chaîne de pile).
- un octet à 0.

CONTENU DU TABLEAU T$ APRES RUN

A	E	I	D
E	F	G	H
I	J	K	L

FORMAT DE STOCKAGE DE T$ EN RAM.

84 128	0 69	2	0 4	0 3

1	0	143	254	0	T$(0,0) =A
1	0	143	250	0	T$(1,0) =E
1	0	143	246	0	T$(2,0) =I
1	0	143	253	0	T$(0,1) =B
1	0	143	249	0	T$(1,1) =F
1	0	143	245	0	T$(2,1) =J
1	0	143	252	0	T$(0,2) =C
1	0	143	248	0	T$(1,2) =G
1	0	143	244	0	T$(2,2) =K
1	0	143	251	0	T$(0,3) =D
1	0	143	247	0	T$(1,3) =H
1	0	143	243	0	T$(2,3) =L

LA PILE OPÉRATIONNELLE

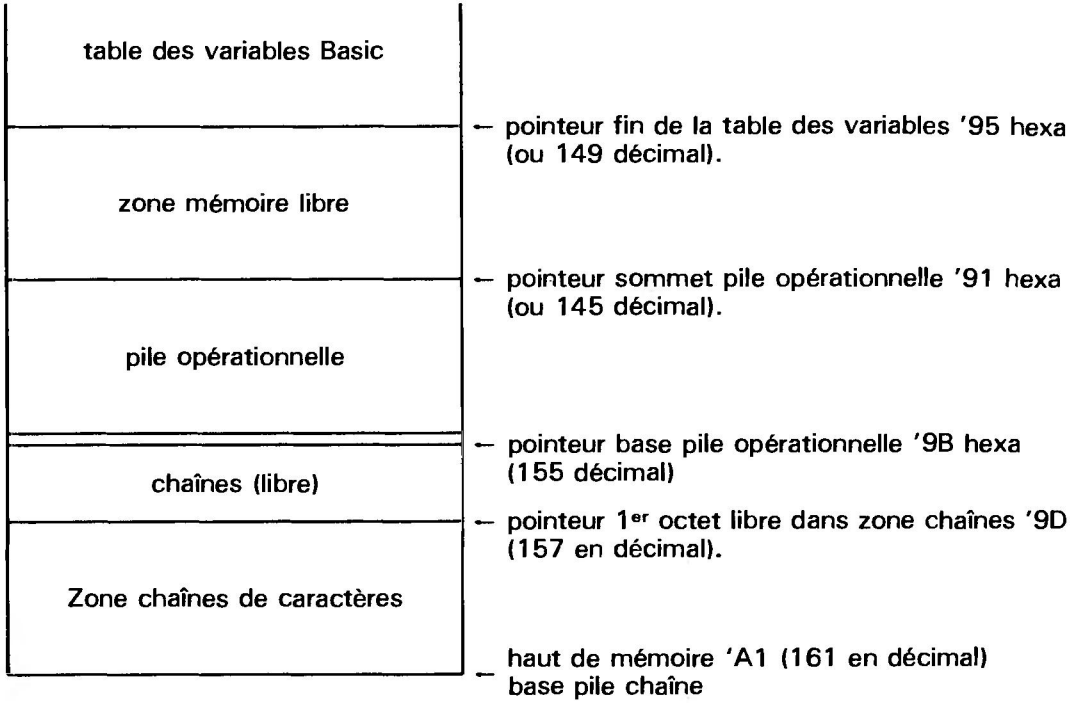
La pile opérationnelle est une zone, située en haut de mémoire vive, dans laquelle le système sauvegarde certaines informations dont il a besoin pour exécuter des instructions de boucle ou de branchement, en Basic en particulier. Ainsi, lors de l'appel d'un sous-programme, est-il nécessaire de mémoriser l'adresse de retour au programme principal. La pile est en fait une mémoire de type LIFO (Last in - first out), c'est-à-dire que c'est le dernier octet entré dans la pile qui en sera sorti le premier). On peut dire que la pile a pour rôle la mémorisation temporaire d'informations nécessaires au système pour gérer correctement un programme à chaque rupture du déroulement normal de ce dernier (sous-programmes, boucles, assembleur, etc...).
De plus, en assembleur les registres du 6803 étant en nombre limité, on utilisera la pile à chaque fois que l'on devra conserver pendant plusieurs cycles le contenu de l'un des registres (voir instruction PSH de l'assembleur 6803). Certaines instructions telles que JSR, BSR, etc... provoquent une mise en pile automatique. La sortie de la pile peut être programmée (voir instruction PUL de l'assembleur 6803) ou provoquée automatiquement par une instruction de retour de sous-programme par exemple. En langage BASIC certaines instructions provoquent une mise en pile automatique : c'est le cas de FOR et de GOSUB, par exemple, qui nécessitent une mémorisation temporaire de certaines informations. La sortie de pile sera alors provoquée par NEXT ou RETURN. La pile est bien entendu repérée par plusieurs pointeurs :

- le pointeur situé à l'adresse '9B hexa (sur deux octets) indique l'adresse de la base de la pile opérationnelle ;
- le pointeur situé à l'adresse '91 hexa (sur deux octets) indique pour sa part l'adresse du sommet de la pile opérationnelle. Notons au passage que lorsque vous demandez à ALICE de vous fournir le nombre d'octets disponibles en mémoire (par MEM en BASIC), ce calcul est effectué à partir du pointeur de fin de la table des variables et du pointeur de sommet de pile, par une différence entre ces deux adresses ;
- enfin, à l'adresse '89 hexa (sur deux octets toujours), on trouve un pointeur sur l'instruction en cours dans la pile.

En fait, la pile n'est pas exactement située en haut de mémoire vive ; en effet, la zone réservée aux chaînes de caractères est placée entre la base de la pile opérationnelle et le haut de la mémoire vive. Par défaut, cette zone a une taille de 100 octets, modifiable sous BASIC par l'instruction CLEAR n, n indiquant le nombre d'octets à réserver aux chaînes alphanumériques. Il est bon d'optimiser la taille de cette zone qui est hélas statique et peut dans certains cas, si elle n'est pas déterminée au plus juste, entraîner une perte de place.

Cette pile réservée aux chaînes de caractères est également repérée par des pointeurs. Ainsi, à l'adresse 'A1 (sur deux octets) se trouve placée l'adresse de la base de la zone réservée aux chaînes de caractères (en général il s'agit du haut de la RAM sauf si on a réservé en haut de mémoire une zone pour le langage machine).
A l'adresse '9D hexa (sur deux octets) on trouve un pointeur indiquant le premier octet libre dans la zone chaînes de caractères de la pile.
Notons, en ce qui concerne la pile opérationnelle, que le sommet est à allocation dynamique, orienté vers le bas de la mémoire et se déplace donc vers la fin de la table des variables. Il est donc utile de faire attention à ces manipulations de pile si l'on veut préserver les informations se trouvant dans la zone mémoire libre.

Schéma général de la pile



Nous allons terminer l'étude de la pile en donnant deux schémas corrrespondant respectivement à l'empilement des informations nécessaire à la gestion d'une boucle FOR et à la gestion de l'appel d'un sous-programme BASIC par l'instruction GOSUB.

Chaque boucle FOR-NEXT active réclame 18 octets dans la pile pour pouvoir s'exécuter.

haut de la pile
(bas de la mémoire)

Code du mot-clé Basic "FOR" : 128 (80H)
OPS de l'adresse de l'index FOR (octet de poids fort) OMS de l'adresse de l'index FOR (octet de poids faible)
Exposant de la valeur du PAS (STEP) OPS1 de la valeur du pas OPS2 de la valeur du pas OPS3 de la valeur du pas OMS de la valeur du pas
Signe du pas (1 si positif, 255 si négatif)
Exposant de la valeur limite maximale à atteindre par l'index OPS1 de la limite maximale à atteindre par l'index OPS2 de la limite maximale à atteindre par l'index OPS3 de la limite maximale à atteindre par l'index OMS de la limite maximale à atteindre par l'index
OPS du n° de la ligne où apparaît le FOR OMS du n° de la ligne où apparaît le FOR
OPS de l'adresse de la première instruction à exécuter dans la boucle OPS de l'adresse de la première innstuction à exécuter dans la boucle

Bas de la pile
(haut de la mémoire)

Chaque GOSUB actif réclame 7 octets dans la pile opérationnelle pour pouvoir s'exécuter.

haut de la pile

Code du mot-clé Basic "GOSUB" = 130 (82H)	
OPS du n° de la ligne où apparaît GOSUB OMS du n° de la ligne où apparaît GOSUB	
OPS de l'adresse de la ligne contenant GOSUB OMS de l'adresse de la ligne contenant GOSUB	
OPS de l'adresse de retour dans le driver d'exécution OMS de l'adresse de retour dans le driver d'exécution	'E5 '42

bas de la pile

L'adresse de retour dans le driver d'exécution (E542 hexa) correspond à l'adresse de retour vers la routine de "RUN" après initialisation des pointeurs. C'est donc l'adresse de la ROM à laquelle on revient pour poursuivre l'exécution du programme.

INITIALISATION

Lorsque vous mettez ALICE sous tension, il se produit un branchement automatique à l'adresse de RESTART que l'on trouve dans les deux derniers octets de la ROM : cette adresse de démarrage à froid du micro-ordinateur est 'F72E (soit 63278 en décimal). Ainsi, pour simuler l'initialisation de votre micro, lorsque vous voulez tout réinitialiser (programme, variables, écran, contrôle, etc...), au lieu de basculer l'interrupteur pour déconnecter l'alimentation, vous pouvez tout simplement taper :

```
EXEC 63278
```

Au bout d'un court instant vous voyez alors apparaître à l'ecran le message :

```
MICROCOLOR BASIC 1.0
COPYRIGHT 1982 MICROSOFT
OK
```

En fait ceci ne produit réellement une réinitialisation complète de la machine que si le contenu de l'adresse 'EA (soit 234 en décimal) est différent de '55 (soit 85 en décimal). Or, si vous avez déjà travaillé sur ALICE, le système a déjà placé cet indicateur à la valeur '55, et vous obtenez alors l'équivalent d'un RESET ou initialisation à chaud, c'est-à-dire que votre programme est conservé. Pour obtenir la réinitialisation complète de la machine il est donc possible d'écrire :

POKE 234,0 : EXEC 63278

Nous allons maintenant, dans les pages qui suivent, détailler le fonctionnement de la procédure d'initialisation d'ALICE de façon à bien montrer comment l'ensemble des fonctions : écran, pointeurs, RAM est généré par le système lors de la mise sous tension. Cette initialisation fait appel à plusieurs procédures réparties dans la ROM à des endroits différents aussi, nous essaierons de suivre cette procédure d'initialisation au niveau logique :

adresse 'F72E (63278) à 'F749

Procédure d'initialisation du micro-ordinateur à la mise sous tension. Place la valeur 'FF à l'adresse 0 qui est le registre de direction du port 1, c'est-à-dire que le port 1 est initialisé en sortie. Ensuite, la valeur 1 est placée à l'adresse '01 et à l'adresse '03 qui sont respectivement le registre de direction du port 2 et le port 2 lui-même : le bit 1 étant le bit sortie Timer. Puis le contenu de l'indicateur situé à l'adresse 'EA (soit 234 en décimal) est testé et, s'il est égal à '55 (soit 85 en décimal), ce qui correspond à une réinitialisation à chaud, le contenu du mot-mémoire dont l'adresse est placée en '4221:'4222 (soit aux octets 16928 et 16929 en décimal) est testé et s'il est égal à 1, un branchement est effectué vers l'adresse contenue dans '4221 et '4222 qui est, après initialisation, **'F7C3.**

- ***adresse F74A à F754***

Dans le cas contraire (adresse 'EA ne contenant pas '55 ou adresse contenue dans '4221:'4222 ne contenant pas 1), initialisation de tous les pointeurs (adresses '80 à '100, c'est-à-dire en décimal 128 à 256) à la valeur 0.

- ***adresse 'F755 à 'F76B***

Détermination du TOP RAM, c'est-à-dire de l'adresse RAM la plus haute par la méthode suivante : prélève le contenu de chaque mot mémoire en commençant à l'adresse '4200 et teste si ce contenu est modifié par l'opération de complémentation : si oui, il s'agit bien d'un mot appartenant à la mémoire vive, sinon, on est arrivé au premier mot de la mémoire morte et dans ce cas l'adresse la plus haute de la mémoire vive est alors placée dans des pointeurs situés en '4250 (soit 16976), 'A1 et '9D (soit respectivment 161 et 157), sur deux octets bien entendu. Ce sont d'ailleurs ces même pointeurs que nous avons été amenés à modifier pour créer une zone en haut de mémoire, réservée au langage machine, c'est-à-dire que par ce procédé nous court-circuitons la procédure située en 'F755.

• ***adresse 'F76C à 'F774***
Réserve les 100 octets du haut de la mémoire pour les chaînes de caractères et place TOP-'064, c'est-à-dire l'adresse du premier octet disponible après la zone réservée aux chaînes, qui constitue la base de la pile opérationnelle en '9B:'9C (c'est-à-dire à l'adresse décimale 155, sur deux octets) et initialise le pointeur de pile avec cette valeur.

• ***adresse 'F775 à 'F77C***
Recopie de 'EB à 'F8 (c'est-à-dire en décimal de 235 à 248) les octets dont les adresses sont comprises entre 'F7D0 et 'F7DD (soit de 63440 à 6346 en décimal).

```
F7D0 7C 00 F5     INC >('00F5)
F7D3 26 03        BNE >'F7D8
F7D5 7C 00 F4     INC >('00F4)
F7D8 B6 00 00     LDA A,('0000)
F7DB 7E E1 C8     JMP >('E1C8)
```

Il s'agit de la procédure permettant l'analyse des instructions BASIC entrées en machine ; le pointeur situé en 'F4 contenant l'adresse du caractère courant dans la table des instructions de programme ou, selon le cas, dans le tampon d'entrée.

• ***adresse 'F77D à 'F784***
Recopie de '4200 à '4230 les contenus des adresses 'F7DF à 'F80F (initialisation des routines d'interruption). En fait cette recopie s'effectue par blocs de trois octets : 00 00 3B, c'est-à-dire deux NOP suivis d'un RTI. Notons que les adresses '4200 à '4300 sont les adresses qui figurent en fin de ROM et correspondent aux adresses vers lesquelles sont effectués les branchements en cas d'interruption

'4200	IRQ2
'4203	IRQ2
'4206	IRQ2
'4209	IRQ2
'420C	IRQ1
'420F	SI
'4212	NMI

• ***adresse 'F785 à 'F791***
Met la valeur '39, c'est-à-dire le code du RTS dans tous les mots dont l'adresse est comprise entre '4285 et '42AE (c'est-à-dire en décimal 17029 à 17070).

• ***adresse 'F792 à F799***
Place la valeur 'FF en '42AF (c'est-à-dire 255 en 17071) et place la valeur '4346 en '93:'94 (c'est-à-dire 16982 en 147:148 en décimal), ce qui correspond à l'initialisation du buffer d'entrée ainsi que de la table des instructions de programme.

• ***adresse 'F79A***
Branchement à la procédure dont l'adresse de début est 'E3CF ; cette procédure réinitialise tous les pointeurs.

• ***adresse 'F79D***
Branchement à la procédure permettant l'affichage d'un écran entièrement vert.

• ***adresse 'F7A0 à 'F7A5***

Affichage du message

MICROCOLOR BASIC 1.0
COPYRIGHT 1982 MICROSOFT

Ce message est prélevé dans une table située en ROM à partir de l'adresse 'F810 (c'est-à-dire 63504 en décimal) et affiché sur l'écran par appel d'une routine d'affichage dont le premier octet est situé en 'E7A8 après avoir placé dans le registre IX l'adresse du premier octet à afficher.

• ***adresse 'F7A6 à 'F7A9***

Replace la valeur '55 dans le pointeur dont l'adresse est 'EA.

• ***adresse 'F7AA***

Branchement au driver d'exécution en 'E271.

• ***adresse 'F7AD à 'F7C2***

Routine de recopie d'une zone mémoire dans une autre zone mémoire, le registre IX contenant l'adresse du premier octet de la zone à recopier et le double accumulateur contenant l'adresse de début de la zone réceptrice. La zone à recopier devra alors commencer par un premier octet contenant le nombre de caractères à recopier.

```
F7AD DD BF      STAD,('BF)
F7AF E6 00      LDA B,('00,X)
F7B1 08         INX
F7B2 A6 00      LDA A,('00,X)
F7B4 DF C1      STX ,('C1)
F7B6 DE BF      LDX ,('BF)
F7B8 A7 00      STA A,('00,X)
F7BA 08         INX
F7BB DF BF      STX ,('BF)
F7BD DE C1      LDX ,('C1)
F7BF 5A         DEC B
F7C0 26 EF      BNE ,'F7B1
F7C2 39         RTS
```

Notons au passage l'utilisation des pointeurs 'BF:'C0 et 'C1:'C2 qui servent de zone de manœuvre pour le stockage temporaire de l'adresse de la zone réceptrice et de la zone à recopier.

• ***adresse 'F7C3 à 'F7CD***

Il s'agit de la routine qui est exécutée en cas de redémarrage à chaud. D'abord l'octet 'E8 est initialisé à 0 (il s'agit de l'indicateur utilisé pour la sortie imprimante ou écran). Puis branchement à la subroutine 'E3EE qui est une réinitialisation des pointeurs et ensuite à la subroutine en 'FBD4 qui efface l'écran avant de se brancher au driver d'exécution en 'E271.

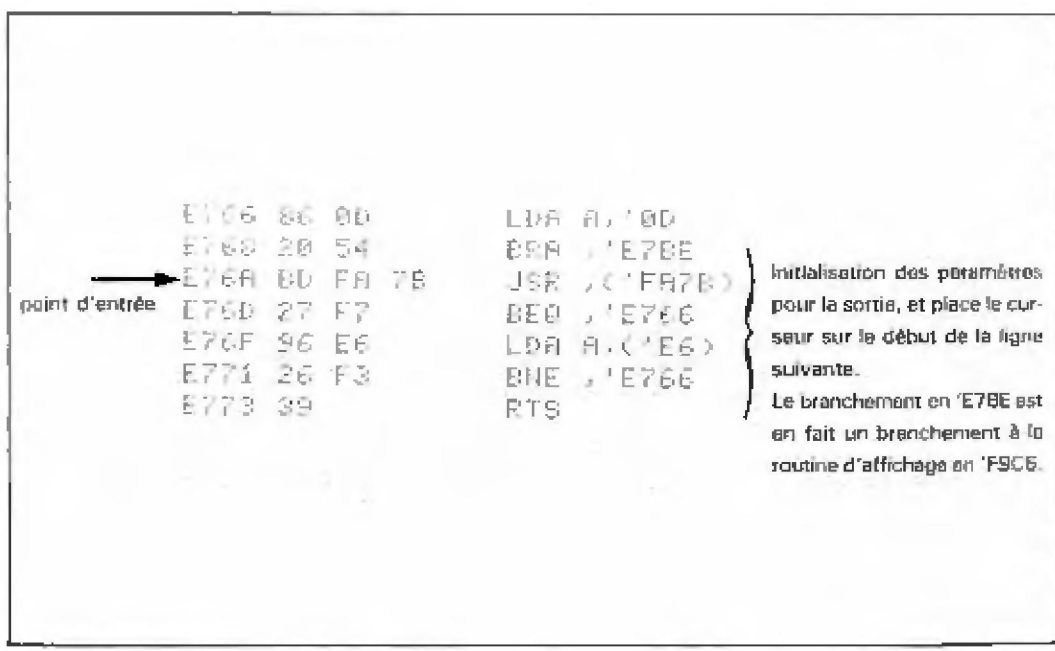
• ***adresse 'E3EE à 'E402***
Recopie de la valeur '4241 à l'adresse '42BD:'42E3 et dépile IX, puis réinitialise le pointeur de pile avec l'adresse contenue dans le pointeur '9B sur deux octets avant d'empiler la valeur 0 et d'initialiser à 0 les octets 'A7, 'A8, '86 et '426E, et enfin d'effectuer un branchement vers l'adresse contenue dans IX.

• ***adresse 'FBD4 à 'FBE4***
Place dans tous les octets correspondant à la zone écran vidéo la valeur '60 (ce qui correspond à un pavé vert) et met à chaque étape le pointeur du dernier caractère affiché à l'adresse '4280 sur deux octets.

EXÉCUTION

• ***adresse 'E271***
A partir de cette adresse, la machine étant initialisée, il devient possible d'entrer des instructions exécutables en mémoire, c'est ce que nous appelerons le driver d'exécution, ce qui ne signifie pas l'exécution immédiate d'un programme BASIC, mais bien l'entrée d'instructions exécutables en mémoire créant ainsi un programme qui sera ensuite recopié dans la table des instructions de programme.

• ***adresse 'E271 à 'E279***
Recopie à l'écran du message OK. Ce message apparaît à chaque fois que la machine est réinitialisée ou que l'exécution d'un programme est terminée et qu' ALICE est en attente d'instructions exécutables. Pour effectuer cette recopie, il y a d'abord un branchement à la routine 'E76A.



```
FA7B BD 42 8B    JSR ,('428B)
FA7E 3C          PSH X
FA7F 37          PSH B
FA80 36          PSH A
FA81 96 E8       LDA A,('E8)
FA83 27 08       BEQ ,'FA8D
FA85 FE 42 27    LDX ,('4227)
FA88 FC 42 29    LDAD,('4229)
FA8B 20 0A       BRA ,'FA97
FA8D F6 42 81    LDA B,('4281)
FA90 C4 1F       AND B,'1F
FA92 CE 10 10    LDX ,'1010
FA95 86 20       LDA A,'20
FA97 DF E4       STX ,('E4)
FA99 D7 E6       STA B,('E6)
FA9B 97 E7       STA A,('E7)
FA9D 32          PUL A
FA9E 33          PUL B
FA9F 38          PUL X
FAA0 39          RTS
```

Sauvegarde des registres dans la pile.

Si 'E8 contient 0 (sortie sur écran), branchement en FA8D

Place dans IX et A:B les 4 octets de '4227 à '422A.

Conserve dans B uniquement les bits 0 à 4 de '4281

et place '20 dans A

Recopie de IX et A:B dans 'E4 à E7.

Restitution des registres.

Cette routine a changé dans ALICE 32 ($FF90 à $FA96) [illegible] les adresses des registres système ($4xxx → $3xxx)

La routine 'FA7B initialise, pour une sortie sur écran, les pointeurs 'E4, 'E5, 'E6 et 'E7 qui contiendront les paramètres de la sortie (numéro de ligne de l'affichage, numéro du caractère sur la ligne, etc...).

- ***adresse 'E274***

IX est initialisé avec l'adresse du premier octet à afficher (ici 'E1BC qui contient '0D, code du retour chariot permettant le passage à la ligne, suivi des codes de O et de K, puis d'un nouveau retour chariot et du drapeau de fin de message : '00). L'affichage est alors demandé par l'appel de la routine 'E7A8.

- ***adresse 'E27A à 'E27E***

Place dans 'E2 et dans 'E3 la valeur 'FF qui signifie que l'instruction en cours de saisie est entrée en mode direct. Lors de la saisie, si l'instruction comporte un numéro, c'est ce numéro qui sera placé dans les octets 'E2 et 'E3.

- ***adresse 'E27F***

Branchement de la subroutine placée en 'FAA4 : il s'agit de l'entrée au clavier de la ligne d'instruction BASIC qui va être stockée dans le tampon d'entrée ; chaque caractère étant affiché à l'écran au moment de sa frappe grâce à un appel de la routine d'affichage 'F9C6.

```
FAA4 BD 42 91    JSR ,('4291)
FAA7 7F 42 7F    CLR ,('427F)
FAAA CE 42 B2    LDX ,'42B2
FAAD C6 01       LDA B,'01
FAAF BD F8 65    JSR ,('F865)
FAB2 7D 00 E9    TST ,('00E9)
FAB5 26 4D       BNE ,'FB04
FAB7 7D 00 E8    TST ,('00E8)
FABA 26 44       BNE ,'FB00
FABC 4D          TST A
FABD 2A 1E       BPL ,'FADD
FABF 7D 42 3A    TST ,('423A)
FAC2 27 19       BEQ ,'FADD
FAC4 BD E4 B2    JSR ,('E4B2)
FAC7 A6 00       LDA A,('00,X)
FAC9 08          INX
FACA 3C          PSH X
FACB 36          PSH A
FACC 84 7F       AND A,'7F
FACE DE 89       LDX ,('89)
FAD0 BD 48       BSR ,'FB1A
FAD2 DF 89       STX ,('89)
FAD4 32          PUL A
FAD5 38          PUL X
FAD6 4D          TST A
FAD7 2A EE       BPL ,'FAC7
FAD9 DE 89       LDX ,('89)
FADB 20 D2       BRA ,'FAAF

FADD 81 0C       CMP A,'0C
FADF 27 C0       BEQ ,'FAA1
FAE1 81 08       CMP A,'08
FAE3 26 08       BNE ,'FAED
FAE5 5A          DEC B
FAE6 27 BF       BEQ ,'FAA7
FAE8 09          DEX
FAE9 BD 37       BSR ,'FB22
FAEB 20 C2       BRA ,'FAAF
FAED 81 15       CMP A,'15
```

Subroutine en '4291 : RTS init. à 0 l'octet tampon clavier.

Prélève caractère au clavier et gère le curseur.

Si le contenu de 'E9 est non nul, branchement en FB04.

Si le contenu de 'E8 est non nul, il s'agit d'une écriture sur écran.

Si code ⩽127, il ne s'agit pas d'un mot-clé.

Teste '423A. Si 0, alors CTRL non enfoncée.

Routine de recherche du mot-clé dans la table des attributs. Si touche CTRL enfoncée, transfert caractère par caractère dans buffer d'entrée du mot-clé Basic tant que bit 7 différent de 1 (le dernier caractère du mot-clé dans la table a toujours le bit 7 à 1).

Boucle.

Si touche CTRL enfoncée, on teste le code : ici gestion du CTRL/Q (code = 08).

```
FAEF 26 0A        BNE ,'FAFB ──────────►
FAF1 5A           DEC B
FAF2 27 B3        BEQ ,'FAA7
FAF4 86 08        LDA A,'08          Gestion de L.DEL.
FAF6 BD F9 C6     JSR ,('F9C6)
FAF9 20 F6        BRA ,'FAF1 ──────────►
FAFB 81 03        CMP A,'03
FAFD 0D           SEC
FAFE 27 05        BEQ ,'FB05 ──────────►
FB00 81 0D        CMP A,'0D
FB02 26 0E        BNE ,'FB12 ──────────►
FB04 4F           CLR A
FB05 07           TPA
FB06 36           PSH A
FB07 BD E7 66     JSR ,('E766)
FB0A 6F 00        CLR ,('00,X)       Lorsque retour chariot.
FB0C CE 42 B1     LDX ,'42B1
FB0F 32           PUL A
FB10 06           TAP
FB11 39           RTS ──────────►
FB12 81 20        CMP A,'20
FB14 25 99        BCS ,'FAAF
FB16 8D 02        BSR ,'FB1A
FB18 20 95        BRA ,'FAAF
FB1A C1 80        CMP B,'80
FB1C 24 F3        BCC ,'FB11
FB1E A7 00        STA A,('00,X)
FB20 08           INX
FB21 5C           INC B
FB22 7E F9 C6     JMP ,('F9C6)
```

• ***adresse 'E282***
Si le bit de retenue (bit C du registre d'état) est à 1, alors c'est que la ligne est entrée en mode direct (ne commence pas par un numéro de ligne). Dans ce cas retour en 'E27A.

• ***adresse 'E284***
Place le contenu du registre d'index en 'F4:'F5 et teste le caractère entré : si le bit C, après comparaison, est à 1, alors il s'agit d'un attribut Basic et un branchement est effectué en 'E293 qui est la routine d'insertion de l'instruction dans la table des instructions de programme.

• ***adresse 'E28D***
Sinon, exécution de la routine 'E311 avant de lancer l'exécution de l'instruction en E53D.

• ***adresse 'E293 à 'E29A***
Lorsqu'un attribut Basic a été rencontré, branchement à la routine 'E6B2 (mise à jour de l'adresse de fin de l'instruction en'A5:'A6) et recopie de l'adresse de chaînage en '42B0:'42B1.

• ***adresse 'E29B***
Branchement à la routine 'E311

• ***adresse 'E29E***
Place la longueur en nombre d'octets de l'instruction saisie en '82.

• ***adresse 'E2A0***
Branchement à la routine 'E3B9 (recherche de l'adresse à laquelle la nouvelle instruction doit être insérée dans la table des instructions de programme. Si l'instruction doit être insérée entre des instructions existantes, le bit C du registre d'état est positionné à 1).

• ***adresse 'E2A3***
Test sur le bit C du registre d'état. S'il est positionné à 1, c'est que l'instruction doit être insérée entre des instructions qui existent déjà, dans ce cas, branchement en 'E2C1 (mise à jour de la table des instructions de programme).

• ***adresse 'E2A5 à 'E2AC***
Sinon, calcul de la nouvelle adresse de début de la table des variables et mise à jour du pointeur de début de la table des variables en '95:'96, puis

• ***adresse 'E2AD à 'E2C0***
Empile le registre d'état et sauvegarde le pointeur de pile en '91:'92 et met le masque d'IT à 1 avant de transférer octet par octet le contenu du tampon d'entrée dans la table des instructions de programme jusqu'à ce qu'on arrive à l'adresse de début de la table des variables. Le pointeur de pile et le registre d'état sont alors restitués.

• ***adresse 'E2C1 à 'E2C5***
Teste le contenu de '42B2. S'il contient la valeur 0, alors branchement en 'E2EB (c'est alors que l'instruction est vide, c'est-à-dire ne comporte rien après le numéro d'instruction).

• ***adresse 'E2C6 à 'E2CF***
Charge le double accumulateur avec l'adresse de début de la table des variables et le recopie en 'BD:'BE, lui ajoute le nombre d'octets de l'instruction à rajouter (contenu de '82) et stocke la nouvelle adresse de début de la table des variables en 'BB:'BC. Ainsi on a entre 'BB et 'BE successivement la nouvelle et l'ancienne adresse de début de la table des variables, stockées dans des zones de manœuvre.

• ***adresse 'E2D0***
Branchement à la routine 'E1FE qui effectue un test de dépassement de capacité mémoire.

• ***adresse 'E2D3 à 'E2DC***
S'il n'y a pas dépassement de capacité, empile le registre d'état et sauvegarde le pointeur de pile en '91:'92 avant de masquer les interruptions et d'effectuer la translation, octet par octet, de la partie du programme qui doit suivre l'instruction à insérer de manière à laisser la place nécessaire à son insertion.

• ***adresse 'E2DB à 'E2EC***
Les octets du buffer d'entrée sont recopiés un par un dans la zone ainsi libérée et les registres sont restitués, puis l'adresse de début de la table des variables est mise à jour en '95:'96, par recopie du contenu de la zone de stockage intermédiaire : 'BB:'BC.

• ***adresse 'E2EB***
Branchement à la routine 'E3D9 (mise à jour des pointeurs).

• ***adresse 'E2EE***
Branchement à la routine 'E2F3 (mise à jour des nouvelles adresses de chaînage après l'adjonction ou la suppression d'une instruction) puis retour en 'E27A.

• ***adresse 'E2F3***
Mise à jour des octets de chaînage de chaque instruction après insertion de toute nouvelle instruction dans la table de programme.

• ***adresse 'E2F3***
Charge dans le registre d'index l'adresse de début de la table des instructions de programme (pointeur '93:94).

• ***adresse 'E2F5 à 'E2F9***
Prélève dans le double accumulateur les deux premiers octets, pointés par le registre d'index (il s'agit des deux premiers octets de l'instruction servant au chaînage) et teste leur contenu. S'ils sont tous les deux à 0, alors il s'agit du drapeau de fin de programme, et dans ce cas, on retourne à la routine d'appel.

• ***adresse 'E2FA à 'E310***
Sinon on empile l'adresse de début de l'instruction contenue dans le registre d'index et on incrémente IX tant que le séparateur avec l'instruction suivante n'est pas rencontré. Lorsque le caractère trouvé dans la table est le code 0, on incrémente IX qui pointe alors sur le premier octet de l'instruction suivante, et on procède, par manipulation de la pile, à la recopie dans le double accumulateur de l'adresse de l'instruction suivante (contenue dans le registre d'index) et dans le registre d'index de l'adresse des octets de chaînage à modifier (précédemment empilée). Dès lors il ne reste plus qu'à mettre à jour les deux octets de chaînage et repositionner IX sur le début de l'instruction suivante avant d'effectuer un branchement en 'E2F5.

• ***adresse 'E311 à 'E323***
Met Ø en '85 et diminue de 1 le pointeur 'F4:F5 contenant l'adresse du caractère courant de l'instruction en cours d'analyse dans la table d'instructions de programme. Initialise le pointeur 'DE:'DF avec l'adresse '42B1 (il s'agit du pointeur contenant l'adresse du dernier octet à afficher dans le buffer de sortie) et sauvegarde le pointeur de pile en '91:'92 ainsi que le registre d'état qui est recopié en '87.

• ***adresse 'E324 à 'E33D***
Après avoir masqué les interruptions par un SEI, teste tous les caractères de l'instruction un par un. Si c'est un blanc, on effectue un branchement en 'E364, sinon on recopie le code prélevé en '81. S'il s'agit d'un guillemet, on effectue un branchement en 'E38D, sinon on teste la valeur contenue en '85 et si elle est différente de 0 on effectue un branchement en 'E364. Si le code prélevé dans l'instruction est celui du point d'interrogation, on le remplace par l'attribut de PRINT et on effectue un branchement en 'E364.

• ***adresse 'E33E à 'E345***
Si le code est inférieur à '30, il s'agit alors de symboles spéciaux et on effectue un branchement en 'E346. Sinon, dans le cas où le code est inférieur à 3C (c'est-à-dire compris entre '30 et '3B), alors il s'agit d'un chiffre ou de symboles : ou ; et dans ce cas on effectue un branchement en 'E364.

• ***adresse 'E346 à 'E348***
Si le code est supérieur à 127, alors il s'agit d'un attribut de mot-clé BASIC et on place

alors dans le registre B qui contenait le code, la valeur '33 avant d'effectuer un branchement en 'E364.

• ***adresse 'E349 à 'E34E***
Initialise le registre d'index avec 'E044 qui est l'adresse de l'octet précédant immédiatement le début de la table des mots-clés dans la mémoire morte et met 0 dans le registre B.

• ***adresse 'E34F à 'E363***
Incrémente IX et prélève les octets l'un après l'autre dans le code source. Tant qu'il s'agit de blancs, on continue seulement à avancer dans le source sans rien faire ; dès que l'on recontre un caractère (ce doit être le premier caractère d'un mot-clé Basic), on recherche ce mot-clé dans la table, dont le premier octet est pointé par IX, par comparaison caractère par caractère, du mot dans le source avec les mots-clés dans la mémoire morte. Dès que le mot-clé est reconnu, place en IX l'adresse du dernier octet à afficher dans le buffer de sortie (pointeur 'DE:'DF).

• ***adresse 'E364 à 'E379***
Mise à jour du pointeur 'F4:'F5 et restitution dans le pointeur de pile de l'adresse précédemment stockée en '91:'92. Restitution également du contenu du registre d'état qui avait été sauvegardé en '87. Le registre d'index est alors incrémenté, puis son contenu est recopié dans le pointeur 'DE:'DF (adresse de fin de sortie dans le buffer). Puis le contenu du registre B (ce registre contenait le code du caractère prélevé dans le source) est recopié dans le buffer. Ainsi, cela a permis de remplacer un mot-clé par son attribut. Si le code était 0 (fin d'instruction), un branchement en 'E3AA est effectué. Si le code contenu dans B était celui du caractère :, alors c'est qu'il y a plusieurs instructions dans la même ligne de programme et un branchement à une routine spécifique, 'E37A, est effectué ; en effet il n'est pas nécessaire, dans ce cas, de mettre à jour les octets de chaînage puisqu'on reste dans la même ligne de programme. Si le code contenu dans B était celui du DATA, un branchement en 'E37A est également effectué alors que dans le cas contraire on passe directement en 'E37C.

• ***adresse 'E37A***
Si on a trouvé dans le registre B le code de : ou de DATA, une recopie du code diminué de '3A est effectuée en '85.

• ***adresse 'E37C à 'E390***
Compare le contenu de B avec le code de REM. En cas d'égalité, la valeur 0 est placée en '81 puis, après positionnement à 1 du masque d'interruption, recopie caractère par caractère la totalité de la ligne d'instruction jusqu'à ce qu'un octet à 0 (fin d'instruction) soit rencontré. Alors un retour en 'E364 est effectué.

• ***adresse 'E392 à 'E3A9***
Routine de recherche d'un mot-clé dans la table des mots-clés située en mémoire morte. Appelée en 'E35B.

• ***adresse 'E3AA à 'E3B8***
Cette routine conclut l'instruction dans le buffer pour initialiser son transfert vers la table des instructions de programme. Deux octets à 0 sont placés à la suite du dernier octet de l'instruction dans le buffer, puis on place dans le double accumulateur la longueur en nombre d'octets de la zone à transférer, c'est-à-dire les deux octets qui vont servir au chaînage, le numéro de ligne, les caractères de l'instruction et le séparateur, et initialise le transfert en plaçant dans le registre d'index IX l'adresse du premier octet à transférer, soit '42B1, adresse de début du buffer qui est recopiée dans le pointeur 'F4:'F5.

• ***adresse 'E3B9 à 'E3CC***
Recherche l'adresse de fin du programme en parcourant la table des instructions de programme selon les octets de chaînage. La fin de la table est atteinte lorsque les octets de chaînage sont à 0. L'adresse de l'octet de drapeau est placée en 'C1:'C2. Si la table est correcte, le bit C du registre d'index est positionné à 1.

• ***adresse 'E3CF à 'E3D8***
Met à 0 les deux octets se situant au début de la table des instructions de programme : leur adresse est prélevée dans le pointeur '93:'94. L'adresse du troisième octet de la table est placée en '95:'96 (pointeur de fin de la table des instructions). Il s'agit de la routine d'initialisation de la table des instructions de programme.

• ***adresse 'E3D9 à 'E3DD***
Place dans le pointeur 'F4:'F5 l'adresse de l'octet précédant immédiatement le début de la table des instructions de programme pour préparer le décodage du programme BASIC et son exécution. Notons que 'E3D9 est l'adresse à laquelle débute la routine de RUN.

• ***adresse 'E3DE à 'E3E0***
Branchement à la routine d'adresse '428E.

• ***adresse 'E3E1***
Initialisation du pointeur contenant l'adresse du premier octet libre de la zone chaîne de caractères dans la pile avec la base de la pile chaîne.

• ***adresse 'E3E5 à 'E3E7***
Branchement à la routine d'adresse 'E560. Il s'agit de la routine associée à RESTORE qui initialise l'adresse du premier octet à prélever dans une liste de DATA (pointeur 'AD:'AE) avec l'adresse de l'octet précédant immédiatement le début de la table des instructions de programme.

• ***adresse 'E3E3 à 'E3ED***
Initialisation des pointeurs concernant les variables : le contenu de '95:'96 (pointeur début de la table des variables) est recopié dans '97:'98 (pointeur de début des variables indicées) et dans '99:'9A (pointeur de fin de la table des variables).

• ***adresse 'E3EE à 'E3F3***
Initialise le pointeur système '423D avec la valeur '4241.

• ***adresse 'E3F4 à 'E3F6***
Place les deux octets en sommet de pile dans le registre d'index et place dans le pointeur de la pile l'adresse de la base de la pile opérationnelle (initialisation des pointeurs de pile).

• ***adresse 'E3F7 à 'E402***
Place 0 en bas de pile ainsi qu'aux adresses 'A7:'A8 (pointeur utilisé par BREAK pour conserver l'adresse de début de l'instruction au cours de laquelle la touche BREAK a été enfoncée ou un STOP a été rencontré), ainsi qu'à l'adresse '86. Puis un branchement est effectué vers l'adresse contenue dans le registre IX d'index (exécution).

Deux routines sont également utilisées lors de l'initialisation de votre micro-ordinateur ALICE. Il s'agit de la routine réalisant la translation d'une partie du programme dans la table des instructions du programme afin de libérer l'espace nécessaire à l'insertion d'une nouvelle instruction, temporairement stockée dans le tampon d'entrée. Cette routine se trouve en 'E1FE. Bien entendu l'exécution de cette routine provoque la mise à jour des pointeurs de début de la table des variables utilisant les zones de manœuvres :
'BB:'BC contiendra l'adresse de l'ancien début de la table des variables.
'BD:'BE contiendra l'adresse de l'ancien début de la table des variables.

'BF:'C0 contiendra l'adresse du premier octet de l'instruction suivant immédiatement celle à insérer, après translation.
'C1:'C2 sera utilisé pour effectuer le déplacement.
Une autre routine importante qui sera très souvent utilisée, tant au cours de l'initialisation que de l'exécution, est la routine de test de dépassement de capacité mémoire. Deux cas peuvent se produire : soit on ajoute une nouvelle instruction BASIC, soit, en cours d'exécution, on rajoute une variable dans la table des variables. Dans le premier cas on exécute la routine depuis l'adresse 'E21E et dans le second cas depuis l'adresse 'E21A.

EXÉCUTION

Le driver d'exécution débute à l'adresse $E53D. Le branchement vers cette routine est effectué obligatoirement par une instruction saisie en mode direct (ce peut être tout simplement RUN...). Le principe en est très simple : par appel à la routine placée en 'EB on prélève caractère après caractère dans la table des instructions de programme en testant à la fois la syntaxe et les codes pour détecter les attributs du Basic. A chaque étape du traitement, c'est-à-dire pratiquement à chaque fois qu'un nouveau caractère est prélevé dans la table des instructions, une exploration du clavier est effectuée grâce à la routine située en 'E566 en vue de détecter un BREAK éventuel ou la frappe d'un shift/@. Si un break est décelé, alors un branchement est effectué à l'adresse 'E57E où les pointeurs permettant de relancer le programme par CONT sont mis à jour.
Dans le cas d'un shift/@, c'est à l'adresse 'E577 que le branchement est effectué. Cette routine boucle sur une exploration du clavier (routine en 'F883) tant que le tampon clavier (adresse '427F) contient 0. On peut d'ailleurs utiliser cette routine en tant que routine de temporisation dans un programme Basic : le programme sera alors stoppé tant qu'aucune touche du clavier ne sera enfoncée.
En cas d'erreur de syntaxe à l'exécution, c'est la routine dont l'adresse est 'E238 qui sera exécutée. Les codes d'erreur sont stockés dans une table en mémoire morte sous forme de deux caractères ASCII utilisés par la BASIC. L'accès à cette table se fait grâce à un index qui est le numéro dans la table du premier caractère du code d'erreur à afficher. Ce numero servant d'index doit être placé dans le registre B et ensuite un branchement à la routine 'E238 doit être effectué. Ainsi :

```
LDA B,'0C
JMP ,('E238)
```

provoquera l'impression du message OM ERROR IN... si un numéro de ligne figure dans le pointeur des numéros de lignes à l'adresse 'E2:'E3. Les codes d'erreurs sont rangés en ROM à partir de l'adresse 'E18A.
Nous allons ci-après donner une liste exhaustive des codes d'erreurs et de leurs significations ainsi qu'une description détaillée de la routine d'affichage de ces codes.

TABLE DES CODES D'ERREUR

Adresse HEXA	Adresse DÉC.	Contenu HEXA	Code erreur	N° erreur décimal	Message complet et signfication
E18A-E18B	57738-57739	4E-46	NF	0	NEXT WITHOUT FOR = un NEXT a été rencontré sans qu'un FOR correspondant n'ait été exécuté (peut correspondre à une interversion de NEXT).
E18C-E18D	57740-57741	53-4E	SN	2	SYNTAX ERROR = erreur de syntaxe résultent le plus souvent d'une mauvaise ponctuation, de parenthèses non fermées, d'un caractère illégal ou d'un mot-clé mal orthographié.
E18E-E18F	57742-57743	52-47	RG	4	RETURN WITHOUT GOSUB = un RETURN a été rencontré alors qu'aucun GOSUB n'est actif.
E190-E191	57744-57745	4F-44	OD	6	OUT OF DATA = une instruction READ a été exécuté alors qu'il n'y evait plus assez de données en DATA pour la satisfaire.
E192-E193	57746-57747	46-43	FC	8	ILLEGAL FUNCTION CALL = appel illégal de fonction. Le programme a tenté une opération sur un paramètre invalide (appel de USR sans POKE de l'adresse, par exemple).
E194-E195	57748-57749	4F-56	OV	10	OVERFLOW = débordement. Une valeur lue ou calculée est trop grande (ou trop petite), pour l'ordinateur.
E196-E197	57750-57751	4F-4D	OM	12	OUT OF MEMORY = espece mémoire insuffisant. Toute la mémoire est soit utilisée, soit protégée. (Ceci peut se produire avec des appels de subroutines ou des FOR-NEXT emboîtés ou encore avec des tableaux très grands).
E198-E199	57752-57753	55-4C	UL	14	UNDEFINDED LINE NUMBER = numéro de ligne non défini. Un branchement est tenté vers une ligne inexistante.

Adresse HEXA	Adresse DÉC.	Contenu HEXA	Code erreur	N° erreur décimal	Message complet et signfication
E19A-E19B	57754-57755	42-53	BS	16	BAD SUBSCRIPT = mauvais indice de tableau Ne correspondant pas à la dimension déclarée.
E19C-E19D	57756-57757	44-44	DD	18	REDIMENSIONNED ARRAY = redéclaration de tableau ayant déjà fait l'objet d'une déclaration implicite ou explicite.
E19E-E19F	57758-57759	2F-30	/0	20	DIVISION BY 0 = tentetive d'exécution d'une division par 0.
E1A0-E1A1	57760-67761	49-44	ID	22	ILLEGAL DIRECT = utilisation de INPUT en mode direct et non dans un programme.
E1A2-E1A3	57762-57763	54-4D	TM	24	TYPE MISMATCH = discordance de type. Tentative d'assignation d'une valeur numérique à une variable alphanumérique ou inversement.
E1A4-E1A5	57764-57765	4F-53	0S	26	OUT OF STRING SPACE = plus d'espace disponible pour les chaînes : demande de plus d'espace réservé aux chaînes qu'il n'en reste réellement.
E1A6-E1A7	57766-57767	4C-53	LS	28	TOO LONG STRING = chaîne trop longue.
E1A8-E1A9	57768-57769	53-54	ST	30	STRING FORMULA TOO COMPLEX = expression trop complexe pour l'ordineteur. La décomposer en expressions plus simples.
E1AA-E1AB	57770-57771	43-4E	CN	32	CAN'T CONTINUE = commande CONT invalide, tentée après une erreur, par exemple.
E1AC-E1AD	57772-57773	49-4F	IO	34	INPUT OUTPUT = erreur d'entrée/sortie avec un périphérique (magnétophone, imprimante, etc...).
E1AE-E1AF	57774-57775	46-4D	FM	36	FILE MISMATCH = erreur avec un fichier (cassette). Tentative de charger un programme en langage machine avec CLOAD ou un programme BASIC avec CLOADM ou un tableau avec CLOAD ou CLOADM.

Le code de l'erreur ayant été placé dans le registre B, un branchement est effectué en 'E238. Dans cette routine d'affichage on place d'abord la valeur 1 dans l'adresse '03, c'est-à-dire que le bit 0 du registre d'entrée/sortie du port 2 est mis à 1. Ensuite le contenu de '426E (soit 17006 en décimal) est testé (notons que cette adresse est utilisée par la sauvegarde sur cassette). Si son contenu est à zéro, alors un appel est effectué à la routine 'E3CF (déjà décrite précédemment, elle remet à jour les pointeurs). Un appel est effectué dans tous les cas à la routine commençant en 'E3EE (déjà décrite également), puis le pointeur 'E8 est mis à 0 (on va donc commander un affichage sur l'écran). Un appel à la routine 'E76A est alors effectué (routine terminant une ligne par le retour chariot) puis à 'E7BC (affichage d'un point d'interrogation sur l'écran). Ensuite, à partir du code erreur contenu dans B, on lance l'affichage du message d'erreur, ainsi que du numéro de ligne dans laquelle s'est produite l'erreur, grâce à la routine 'E7A8 permettant d'afficher une chaîne de caractères.
Cette routine que vous pourrez désassembler à l'aide du programme désassembleur qui se trouve en fin du présent ouvrage est extrêmement intéressante dans la mesure où elle montre la façon dont peut être réalisé en langage machine l'affichage d'une chaîne de caractères. Elle fait bien sûr appel à la routine 'F9C6 d'affichage d'un caractère dont le code ASCII a auparavant été placé dans l'accumulateur A. Par ailleurs, on peut noter que les pointeurs d'arrêt n'étant pas remis à jour lors de l'apparition d'une erreur, il n'est pas possible de relancer directement (par CONT, par exemple) l'exécution d'un programme depuis le point où il s'est arrêté. Nous étudierons l'ensemble des routines spéciales dans les pages qui suivent, parmi lesquelles la routine 'F9C6 d'affichage d'un caractère utilisée ici.
Avant d'aborder l'aspect du langage BASIC d'ALICE, nous allons donc donner quelques accès à des routines particulièrement intéressantes de la mémoire morte. Cette étude n'a pas la prétention d'être exhaustive, aussi seuls quelques exemples de routines importantes seront abordés, ainsi que dans certains cas les moyens de redresser une erreur de manipulation.

ADRESSES IMPORTANTES

• ***adresse 'E238***
Affichage des messages d'erreurs. Avant de lancer l'exécution de cette routine, il est nécessaire de placer dans le registre B le numéro d'erreur dont on veut faire afficher le message. Si le pointeur 'E2:'E3 contient 'FFFF, alors le message XX ERROR sera affiché ; sinon, si 'E2:'E3 contient une valeur sur deux octets YY, alors le message XX ERROR IN YY sera affiché.

• ***adresse 'E57E***
Contrôle de la touche BREAK. Lorsque la touche BREAK a été enfoncée, cette routine met à jour tous les pointeurs pour un redémarrage à chaud du programme et provoque l'affichage du message : BREAK ou BREAK IN YY selon que 'E2:'E3 contient 'FFFF ou non.

• ***adresse 'E404***
Sélection de l'imprimante pour effectuer une sortie. Cette routine place la valeur 'FE dans l'indicateur 'E8. A l'affichage un test est effectué sur 'E8. S'il contient 'FE (en fait s'il contient une valeur non nulle...), alors la sortie est effectuée sur imprimante, et dans le cas contraire (si 'E8 contient 0), la sortie est effectuée sur écran.

• ***adresse 'E577***
Interruption de l'exécution jusqu'à ce qu'une touche du clavier soit enfoncée. Il s'agit de la routine de controle de shift/@. Elle consiste en une boucle d'exploration du clavier dont la condition de sortie est : contenu de '427F non nul ('427F est le tampon clavier, il contient le code ASCII du caractère frappé au clavier). Elle poura être utilisée en temporisation.

• ***adresse 'EBC7***
Transforme un nombre stocké en virgule flottante en machine, en nombre décimal pour l'affichage en utilisant les octets de 'C9 à 'CD dans la zone système.

• ***adresse 'ECE3***
Routine permettant de transformer un entier sur seize bits (compris entre 0 et 'FFFF), contenu dans le double accumulateur A:B en entier signé, sous forme d'une suite de chiffres pour affichage (voir exemple dans routine de démonstration sur USR).

• ***adresse 'EF4C***
Prélève dans le buffer, ou dans une zone définie par le programmeur, un nombre compris entre 0 et 65535 (décimal) en un entier sur seize bits, résultat dans le registre d'index IX.. Si le nombre est décimal, la routine conserve uniquement sa partie entière pour faire la transformation.

• ***adresse 'E7A8***
Permet d'afficher à l'écran (ou sur imprimante par appel de 'E404) une ligne dont l'adresse du premier octet devra être placée dans le registre d'index IX. Le message à afficher devra se terminer par un octet à 00 servant de drapeau de fin. Il est possible de placer dans ce message des octets à '0D (code du retour chariot) si l'on souhaite imprimer le message sur plusieurs lignes d'écran. Attention toutefois à la longueur en nombre d'octets de la zone à afficher : elle ne doit pas excéder 255. En effet cette routine commence par calculer la longueur de la zone à afficher et la place dans le registre B.
On pourra aussi, c'est une autre façon de l'utiliser, placer dans B le nombre d'octets à afficher, dans IX l'adresse du premier octet à afficher et appeler la routine à l'adresse 'E7AE.

• ***adresse 'F72E***
Initialisation du système à la mise sous tension. Il s'agit d'une réinitialisation complète du micro-ordinateur (démarrage à froid).

• ***adresse 'F76C***
Réinitialisation à chaud du système. Cette routine sera utilisée lors d'une réinitialisation en préservant une zone mémoire destinée aux instructions en langage machine (voir à cet effet le programme permettant de réserver en haut de mémoire vive une zone pour le langage machine).

• ***adresse 'F847***
Cette routine permet de changer le curseur. Il suffit de placer dans le registre A le code du curseur que l'on souhaite obtenir et d'appeler la routine. Vous pouvez, par exemple, essayer :

```
LDA A,'86
JSR ,'F847
```

• ***adresse 'F883***
Attente d'un caractère au clavier. Le caractère frappé au clavier sera alors disponible en '427F et dans l'accumulateur A. En fait on disposera directement de son code ASCII. Tant qu'aucun caractère n'est entré au clavier, on a la valeur 0.

• ***adresse 'F9C6***
Affichage d'un caractère à la position courante du curseur. Le code du caractère à afficher doit être placé auparavant dans l'accumulateur A. La position du curseur peut être modifiée en agissant sur les adresses '4280 et '4281 qui contiennent, sur deux octets, l'adresse de la position courante du curseur dans la mémoire vidéo.
Par exemple :
LDA A,'40
STA A,('4280)
LDA A,'80'
STA A,('4281)
LDA A,'41
JSR ,'F9C6
permettra d'afficher le caractère 'A' en première colonne de la 5e ligne de l'écran.
Rappelons que la mémoire vidéo s'étend de '4000 à '4200 et que chaque ligne d'écran peut contenir 32 caractères.

• ***adresse 'F861***
Temporisation d'une valeur spécifiée dans le registre d'index IX. Ce qui permet, IX étant un registre sur seize bits, de placer dans IX des valeurs entre 0000 et 'FFFF, durée maximale de la temporisation.

• ***adresse 'FA64***
Scrolling vertical. Cette routine permet de faire remonter le contenu de l'écran d'une ligne, la première ligne étant recouverte par la suivante et la dernière ligne de l'écran étant mise à blanc.

• ***adresse 'FAA4***
Attente d'une ligne au clavier. Chaque caractère entré (correspondant à une touche enfoncée) est stocké dans un buffer dont l'adresse est fournie dans 'AF:'B0. L'entrée continue jusqu'à ce qu'un retour chariot ou la touche "BREAK" ne soit utilisée, ou que le buffer soit plein.

• ***adresse 'FBD4*** à $F8E4
Effacement complet de l'écran (CLS en BASIC).

• ***adresse 'FBD6***
On peut utiliser cette routine pour remplir l'écran avec un caractère donné. Le code de ce caractère devra au préalable avoir été placé dans le registre B. Ainsi, par exemple :
LDA B,'FF
JSR ,'FBD6
couvrira l'écran de pavés oranges.

• ***adresse 'FCC0***
Écriture d'un bloc sur la cassette, l'adresse de début du bloc à transférer étant placée en '426F:'4270, l'adresse de fin du bloc en '4271:'4272 et la longueur du bloc en '426C:'426D. L'octet d'adresse '4267 indique le type de bloc dont il s'agit :
0 — Basic
2 — Code machine
4 — Tableau

• ***adresse 'FCB7***
Écriture amorce bande sur la cassette.

• ***adresse 'FE46***
Lecture de l'en-tête sur la cassette. Lorsque le bloc correspondant au titre entré est trouvé, affichage de la lettre "F" en haut et à gauche de l'écran.

• ***adresse 'FEB9***
Routine de lecture d'un bloc sur cassette.

• ***adresse 'FF14*** à FF4D
Lecture d'un octet sur cassette. L'octet lu est transmis dans le registre accumulateur B.

• ***adresse 'FF4E***
Écriture d'un octet série sur la cassette. Le contenu du registre accumulateur B est envoyé en série sur la bande.

• ***adresse 'FFAB***
Générateur d'un son sur le haut-parleur du téléviseur. Le registre A devra au préalable avoir été chargé avec le numéro du son à émettre et le registre B avec la durée de ce son. Par exemple, pour obtenir la note "LA" on peut exécuter la séquence suivante :

```
LDA A,'A5
LDA B,'40
JSR ;('FFAB)
```

• ***adresse 'C9***
Adresse du premier octet de la zone dans laquelle est effectuée la transformation des nombres en virgule flottante.

• ***adresse 'EB***
Routine d'exploration des programmes BASIC, caractère par caractère.

TABLEAU DES NOTES DE MUSIQUE

Note	Octave							
	Première		Deuxième		Troisième		Quatrième	
	hexa	déc.	hexa	déc.	hexa	déc.	hexa	déc.
DO	—	—	66	102	B4	180	DB	219
DO #	—	—	6F	111	B9	185	DD	221
RÉ	—	—	77	119	BD	189	DF	223
RÉ #	—	—	7F	127	C1	193	E3	227
MI	0B	11	87	135	C4	196	E5	229
FA	19	25	8D	141	C8	200	—	—
FA #	25	37	9E	148	CB	203	—	—
SOL	32	50	9A	154	CE	206	—	—
SOL #	3E	62	A0	160	D1	209	—	—
LA	48	72	A5	165	D4	212	—	—
LA #	53	83	AB	171	D6	214	—	—
SI	5D	93	AF	175	D9	217	—	—

Bien sûr il existe dans la ROM d'ALICE beaucoup d'autres routines très intéressantes que vous pourrez retrouver en désassemblant à l'aide du petit programme désassembleur donné à la fin de cet ouvrage, mais il est impossible ici de les énumérer toutes. Celles qui sont présentées dans le tableau précédent sont, à mon avis, les plus importantes car elles peuvent facilement être utilisées par la plupart des programmes BASIC ou en assembleur. De plus, leur déassemblage vous amènera sans aucun doute à trouver beaucoup de choses encore, dans la mesure où chacune d'entre-elles fait appel elle-même à plusieurs sous-routines.
Pour finir l'étude de la ROM, intéressons-nous maintenant aux routines associées aux attributs BASIC. On peut, en fait, distinguer trois grands types de routines :

- celles qui sont associées à des mots-clés du Basic (leur attribut est inférieur à 'A1, soit 161 en décimal) ;
- les routines arithmétiques (leur attribut est compris entre 'A7 et 'AD, soit entre 167 et 173 décimal) ;
- les routines associées à des fonctions (leur attribut est compris entre 'B1 et 'C8, soit entre 177 et 200 en décimal).

On constate que pour les attributs compris entre 'A2 et 'A6 et entre 'AD et 'B0, soit (TAB(, TO, THEN, NOT, STEP...) on ne définit aucun routine. En effet, ces attributs sont toujours associés à d'autres attributs ayant leur propre routine. Quant à 0FF, bien que figurant dans la table, nous n'avons trouvé dans la ROM aucune routine à lui associer.
Étudions en premier lieu les routines associées à des mots-clés BASIC. Dans la mémoire morte du micro-ordinateur on trouve une table contenant les adresses des routines associées à ces mots-clés. Cette table est mise en œuvre par le programme d'exécution d'instruction qui vient rechercher l'adresse de la routine associée au mot-clé correspondant. Chaque adresse de routine est positionnée dans la table, suivant un index de position identique à celui de l'attribut dans la table de création des attributs. La table des adresses de routines commence en 'E148. Le principe du branchement est extrêmement simple : on commence par placer dans le registre B l'indice du mot-clé dans la table de création, puis dans le registre IX d'index, la valeur 'E148. Ensuite par ABX on incrémente le registre IX du contenu du registre B avant de procéder à un branchement : JMP ,('00,X). Nous allons donc, dans cette première partie, commencer par présenter la table des routines associées aux mots-clés, assortie de quelques commentaires sur ces routines. On trouvera une table générale de toutes les routines à la fin de ce chapitre.

Adresse hexa de la routine	Mot-clé	Position dans la table (hexa)	Rôle
'E5B8	CLEAR	15	CLEAR seul, non suivi d'un nombre provoque immédiatement en 'E3DE la remise à 0 de toutes les variables ainsi que la réservation de 100 octets pour les chaînes. CLEAR suivi d'un nombre réinitialise les variables et réserve la moitié de l'espace indiqué par ce nombre.
'FD5C	CLOAD	17	Charge un fichier à partir d'une cassette : le premier venu s'il n'y a pas de nom après CLOAD, sinon parcourt la cassette jusqu'à ce qu'un fichier portant le même nom soit trouvé. Le nom a au maximum 8 caractères. Le nom du ficher à rechercher est stocké de '4257 à '425E et celui du programme trouvé sur la cassette est chargé entre les adresses '425F et '4266. Le nombre d'octets composant le nom est pour sa part stocké en '4256. Les fichiers peuvent être de trois sortes : - programme BASIC. Dans ce cas l'octet d'adresse '4267 sera mis à 0 et l'appel se fera par CLOAD nom ; - tableau numérique à une dimension. Dans ce cas l'octet d'adresse '4267 sera mis à 4 et l'appel se fera par CLOAD* nom de tableau, nom de fichier ; - code machine. Dans ce cas l'octet d'adresse '4267 sera mis à 2 et l'appel se fera par CLOADM ''nom de fichier'', adresse début - nombre d'octets. Dans ce cas le chargement en mémoire centrale s'effectuera sur le nombre d'octets indiqués à partir de l'adresse spécifiée.
'FBBF	CLS	1D	Un test est d'abord effectué pour savoir si CLS est suivi d'un nombre. Si on a CLS seul, un branchement est effectué en 'FBD4, routine qui remplit l'écran avec des pavés verts. Dans le cas où CLS est suivi d'un argument, cet argument est transformé en nombre entier sur un octet qui est placé dans le registre B. Si le contenu de B est supérieur à 8, un CLS normal est exécuté et un branchement à la routine d'affichage d'une ligne est effectué en initialisant cet affichage à l'adresse 'F833, c'est-à-dire au mot

Adresse hexa de la routine	Mot-clé	Position dans la table (hexa)	Rôle
'FBBF suite			"MICROSOFT", résultat pour le moins curieux ! Dans le cas où le contenu de B est inférieur ou égal à 8, l'écran est effacé dans la couleur spécifiée par la valeur. On peut aussi avoir CLS (expression ou identificateur) ; dans ce cas les parenthèses sont obligatoires.
'E5A6	CONT	13	Si le pointeur sur l'adresse du premier octet de l'instruction à exécuter ('A7:'A8) contient 0, on affiche le message "CN ERROR", sinon on recopie cette valeur en 'F4:'F5 et le numéro d'instruction est mis à jour avant de relancer le programme sur la ligne où il avait été arrêté par un "BREAK", les instructions STOP ou END, ce qui correspond à des interruptions volontaires.
'FC3B	CSAVE	18	Sauvegarde un fichier sur cassette. Le nom doit avoir au maximum huit caractères. Deux possibilités sont offertes : - sauvegarde d'un programme BASIC par CSAVE "nom" ; - sauvegarde d'un tableau numérique à une dimension par CSAVE* nom du tableau, "nom de fichier". Par contre, la possibilité de sauvegarder du code machine n'est pas prévue. On peut y remédier en entrant le petit programme ,donné par ailleurs dans cet ouvrage, que l'on exécutera à chaque fois qu'un CSAVEM serait nécessaire. Les zones système utilisées sont les mêmes que pour CLOAD.
'E651	DATA	05	DATA enregistre des données séquentiellement dans un programme.
'EB12	DIM	0B	Déclare un ou plusieurs tableaux. Après vérification de la syntaxe, une recherche est effectuée dans la table et si la variable est trouvée, affichage du message "DD ERROR" sinon réservation.
'E57F	END	09	Arrêt de l'exécution du programme. Notons qu'un programme peut être relancé, dans le cas d'ALICE, par CONT lorsque END n'est pas la dernière instruction du programme (voir STOP).
'FC04	EXEC	1F	EXEC doit être suivi d'un argument qui est l'adresse à partir de laquelle on veut exécuter un sous-programme en langage machine. L'argument

Adresse hexa de la routine	Mot-clé	Position dans la table (hexa)	Rôle
'FC04 suite			de EXEC est d'abord transformé en nombre entier sur deux octets et sauvegardé en '421F:'4220 puis un branchement est effectué à l'adresse ainsi calculée. Attention à prévoir un RTS en fin de votre routine et, si vous voulez revenir dans de bonnes conditions au BASIC, à sauvegarder les registres...
'E4C4	FOR...TO	00	Détermine les bornes d'une itération qui se fera à partir d'une valeur initiale jusqu'à une valeur finale. A la rencontre de cette instruction, le système sauvegarde dans la pile les paramètres concernant ce FOR (voir chapitre sur la pile opérationnelle).
'E604	GOSUB	02	Appel d'un sous-programme qui commence à la ligne dont le numéro est spécifié après GOSUB. A la rencontre de cette instruction le système sauvegarde dans la pile les paramètres lui permettant d'effectuer un retour au programme principal (voir chapitre sur la pile opérationnelle). Avant cette opération un test est effectué pour vérifier que l'espace mémoire disponible est suffisant.
E61A	GOTO	01	Provoque une rupture de séquence et un branchement inconditionnel à l'instruction dont le numéro suit GOTO. Un test est effectué sur la table des instructions de programme pour trouver la ligne sur laquelle doit s'effectuer le branchement. La recherche s'effectue grâce aux octets de chaînage. Si on arrive en fin de table sans trouver le numéro de ligne spécifié, un message "UL ERROR" est affiché. Sinon, on positionne les pointeurs 'F4:'F5 et les pointeurs concernant l'instruction de destination et le programme se poursuit.
'E672	IF...THEN	04	L'instruction qui suit THEN est exécutée seulement si la condition précédant THEN est vraie. Notons qu'il est possible d'avoir plusieurs conditions reliées par des opérateurs logiques, de même que plusieurs instructions séparées par : peuvent exister après le THEN.
'E7DE	INPUT	08	Un test est d'abord effectué pour vérifier que INPUT n'est pas utilisé en mode direct ('E2:'E3 ne doit pas contenir la valeur 'FFFF) sinon un

Adresse hexa de la routine	Mot-clé	Position dans la table (hexa)	Rôle
'E7DE suite			message ''ID ERROR'' est affiché. Puis le caractère suivant est testé. S'il s'agit d'un guillement, le message suivant le guillemet est affiché jusqu'au guillemet suivant qui doit être suivi d'un point virgule. Un point d'interrogation est alors affiché si la syntaxe est correcte, et le clavier est testé (routine 'FAA4) en attente des données, lesquelles sont traduites dans le format machine et stockées en mémoire. Lorsque le nombre de données entrées est supérieur au nombre attendu, un branchement est effectué à la routine d'affichage initialisée à l'adresse 'E8AB: ''EXTRA IGNORED''. Le nombre entré est traduit en virgule flottante par les routines 'F359 et 'F270. Lorsqu'il s'agit d'une chaîne, elle est placée dans la zone chaîne de pile.
'E6D3	LET	0D	Prélève d'abord l'identificateur de la variable et vérifie qu'il est suivi du signe ''='', puis évalue l'expression à droite du signal égal, vérifie qu'elle est de même type que la variable à laquelle elle est affectée. La variable est alors recherchée dans la table des variables et si elle n'existe pas elle est créée, puis la valeur lui est attribuée ('E705).
'E40D	LIST	14	Si LIST est suivi d'un argument, celui-ci est évalué (branchement en 'E6B2) puis branchement à la routine 'E3B9 pour déterminer les bornes entre lesquelles on doit lister. Puis affichage depuis la borne initiale, jusqu'à la borne finale du contenu de la table des instructions de programme, en effectuant un décodage des attributs.
'E40B	LLIST	19	Avant d'effectuer le LIST on effectue un branchement à la routine 'E404 de sélection de l'imprimante, puis le LIST est effectué avec l'indicateur 'E8 contenant 'FE.
'E71C	LPRINT	1A	Avant d'effectuer le PRINT on exécute la routine 'E404 de sélection de l'imprimante, puis le PRINT est effectué avec l'indicateur 'E8 contenant 'FE.
'E3CF	NEW	16	Cette fonction efface théoriquement le contenu de la mémoire, libérant celle-ci pour entrer un nouveau programme. En fait la mémoire n'est pas effacée, le NEW ne fait que réinitialiser certains pointeurs. Donc, en cas de NEW malencontreux,

Adresse hexa de la routine	Mot-clé	Position dans la table (hexa)	Rôle
'E3CF suite			il suffit de repositionner correctement les pointeurs pour retrouver le programme. La routine de NEW commence par mettre à Ø les deux octets de chaînage de la première instruction. Ces deux octets contiennent en principe l'adresse du premier octet de la deuxième instruction. En rétablissant la valeur initiale vous pouvez de nouveau LISTER le programme. Pour retrouver cette adresse vous pouvez, par exemple, exécuter en mode direct l'instruction suivante : FORI = 17222 TO 17350 : PRINT I,PEEK(I) : NEXT I Il suffit alors de rechercher la valeur de I suivant immédiatement le séparateur d'instruction, c'est-à-dire le code Ø et de la coder sur deux octets en calculant : INT(I/256) et I – 256*(INT(I/256) puis de POKER ces deux valeurs respectivement en 17222 et 17223. Ensuite listez votre programme et ô surprise, il réapparaît. Toutefois, il n'est plus exécutable ; en effet, les pointeurs sur la table des variables ont également été réinitialisés. Il faut donc leur affecter l'adresse suivant le dernier octet du programme. Bien sûr il n'est pas question de faire un programme BASIC vous permettant de résoudre ce problème, car bien entendu vous effaceriez ainsi le programme que vous cherchez à retrouver. Alors, écrivez donc une routine en assembleur parcourant tous les octets de chaînage jusqu'à ce que vous trouviez les deux octets de chaînage à 0. L'adresse suivant ce deuxième octet à 0 est celle que vous cherchez, et il suffit alors de la placer en '95:'96, '97:'98 et '99:'9A. Si l'assembleur vous effraie, vous avez une autre possibilité à partir de BASIC, mais exclusivement en mode direct. vous venez de retrouver n l'adresse de début de la deuxième instruction. Écrivez alors : PRINT PEEK(n)*256+PEEK(n) vous aurez alors l'adresse de début de la troisième instruction, etc... Jusqu'à ce que vous obteniez 0, alors reprenez la dernière valeur obtenue non nulle, ajoutez-lui 2 et transformez la sur deux

Adresse hexa de la routine	Mot-clé	Position dans la table (hexa)	Rôle
'E3CF suite			octets comme vu précédemment et POKEZ en 149:150, 151:152, 153:154. Ensuite tentez un RUN : votre programme est de nouveau exécutable !
'E692	ON	07	ON...GOSUB et ON...GOTO sont traités dans la même routine. Un test est d'abord effectué pour détecter les erreurs de syntaxe et le numéro de ligne vers lequel le branchement doit être effectué est calculé dans la liste, puis, une fois ce numéro de ligne déterminé, on se branche à la routine du GOSUB ou du GOTO.
'EF66	POKE	12	Place à l'adresse spécifiée comme premier argument la valeur entière sur un octet spécifiée comme deuxième argument.
'E71F	PRINT	06	Un premier test est effectué pour savoir si au moins un caractère suit le print. Sinon on place le code du retour chariot dans A et on exécute la routine d'affichage 'F9C6. Si un caractère suit PRINT, on teste s'il s'agit du @ et dans ce cas, en 'E72B jusqu'à 'E735 on détermine la position à laquelle doit se faire l'affichage. On teste alors si le caractère prélevé est le code de TAB(, auquel cas un branchement est effectué à la routine de TAB(, en 'E78C. Sinon, on compare au code de la virgule et, en cas d'égalité, on lancera l'impression avec positionnement en 'E774. Si le code est celui du point virgule, c'est la routine 'E7A2 qui est appelée. Les caractères à afficher seront successivement placés dans le buffer de sortie, puis la ligne sera imprimée en une seule fois.
'E80E	READ	0C	Les pointeurs sont mis à jour puis la routine 'E816 jusqu'à 'E827, commune à READ et à INPUT, est exécutée et un branchement est effectué en 'E877 où la valeur à entrer est recherchée dans les listes de DATA grâce aux pointeurs sur les DATA en cours. Puis, enfin, branchement en 'E82E est effectué.
'E685	REM	03	Tout ce qui suit REM est ignoré. On s'en sert seulement pour placer des commentaires dans le programme. Cela ne fait que provoquer un avancement du pointeur 'F4;'F5 sur le début de l'instruction suivante.

Adresse hexa de la routine	Mot-clé	Position dans la table (hexa)	Rôle
'FB55	RESET	1C	Efface le point de position (x, y) sur l'écran.
'E560	RESTORE	0F	Réinitialise le pointeur 'AD:'AE sur la file des DATA avec l'adresse de début de la table des instructions de programme.
'E631	RETURN	10	Recherche dans la pile les paramètres du dernier GOSUB appelant. S'il n'y a pas de GOSUB dans la pile, alors un message "RG ERROR" est affiché. Sinon, l'adresse de retour est retrouvée, les pointeurs sont remis à jour et les paramètres concernant le GOSUB appelant sont enlevés de la pile opérationnelle et le programme se poursuit.
'E5FA	RUN	0E	Un test est effectué pour savoir si RUN est suivi d'un numéro de ligne. Si oui, un branchement est effectué en 'E3D9. Si non, un appel est fait à la routine 'E3DE et le programme est exécuté.
'FB25	SET	1B	Des tests sont effectués sur la syntaxe et la détermination du point à afficher est effectuée par la routine commençant en 'FB40.
'FE2F	SKIPF	20	La subroutine 'FDA2 d'initialisation de l'en-tête est exécutée jusqu'à ce que le programme cherché soit trouvé sur la cassette.
'FFA5	SOUND	1E	Cette routine permet d'émettre un son de hauteur comprise entre 0 et 255 pendant une durée comprise entre 0 et 255, ceci en envoyant des impulsions sur la sortie d'écran par le bit 7 du registre de contrôle du MC6847 en 'BFFF.
'E57E	STOP	11	Lorsque la touche "BREAK" est enfoncée ou lorsque un STOP est rencontré, le bit C du registre d'état est positionné à 1, puis les pointeurs permettant la reprise de l'exécution du programme sont mis à jour. Puis le bit C du registre d'état est testé : s'il est à 0, c'est que l'instruction traitée était END et le message OK est affiché. S'il est à 1, c'est qu'on a à faire à "BREAK" ou STOP et le message BREAK IN XX est affiché.

Après les routines associées aux commandes du BASIC, on peut évoquer très simplement les routines concernant les opérateurs mathématiques.
Elles sont au nombre de 7 : les cinq opérations arithmétiques +, −, *, / et ↑ et les deux opérateurs logiques AND et OR. On verra dans un prochain tableau les priorités qui les caractérisent et permettent d'évaluer les expression.

Nous nous contenterons ici de donner les adresses des routines associées à ces opérateurs.
Pour mieux comprendre leur fonctionnement, il vous sera possible de désassembler la ROM à partir des adressees indiquées.

Adresse hexa	Opérateur	Priorité	Rôle
'EF80	+	79	Addition virgule flottante.
'EF75	–	79'	Il s'agit en fait d'une addition en virgule flottante pour laquelle le deuxième opérande est complémenté.
'F0F1	*	7B	Multiplication virgule flottante.
'F1C8	/	7B	Division virgule flottante.
'F556	↑	7F	Élévation puissance.
'EA8E	AND	50	Et logique.
'EA8D	OR	46	Ou logique.

Nous terminons l'étude des routines associées aux attributs de BASIC avec l'étude des fonctions BASIC.
Ces fonctions, à l'inverse des commandes étudiées précédemment, sont des mots-clés BASIC qui ne peuvent apparaître qu'à droite d'un signe égal. Le principe est le même que pour les commandes : la table des adresses des routines associées aux fonctions commence en 'E000 et chaque adresse est positionnée dans la table suivant un index de position identique à celui de l'attribut correspondant. Les attributs des fonctions sont compris entre 'B1 et 'C8. A l'exécution, lorsqu'une instruction est décodée, un test est effectué sur le code du caractère prélevé dans la table des instructions de programme. Si ce code est compris entre 'B1 et 'C8, alors il s'agit d'une fonction (la routine qui effectue ce test commence en 'E92E et lorsque l'attribut est celui d'une fonction, un branchement est effectué vers 'E949). Le traitement des fonctions est exécuté par la routine dont l'adresse de début est 'EA55. Cette séquence effectue des tests permettant de séparer les fonctions en quatre groupes.

- les fonctions numériques à un argument qui seront traitées en 'EA7F ;
- les fonctions sans argument (INKEY$, MEM) traitées en 'EA81 ;
- les fonctions alphanumériques et à plusieurs arguments traitées en 'EA62 ;
- et enfin en 'EA69 les fonctions POINT et VARPTR.

Dans tous les cas, une vérification est effectuée sur la syntaxe, puis le registre d'index est chargé avec l'adresse de la routine prélevée dans la table et un branchement est effectué vers cette routine de traitement.

Examinons maintenant, selon le même mode que les commandes BASIC, la table des routines associées en fonction BASIC.

Adresse hexa de la routine	Mot-clé	Position dans la table (hexa)	Rôle
'F2D5	ABS	02	Calcule la valeur absolue de l'argument, c'est-à-dire indépendante du signe. Cette routine positionne l'octet d'adresse 'CE à 0.
'EEA2	ASC	0F	Renvoie le code ASCII du premier caractère de la chaîne constituant l'argument, c'est-à-dire la valeur de l'octet correspondant au premier caractère de la chaîne dans la zone source.
'EE8E	CHR$	10	CHR$(n) renvoie le caractère ou la commande correspondante à n dans le code ASCII.
'F686	COS	09	COS(x) fournit le cosinus d'un angle x mesuré en radians.
'F5C9	EXP	07	EXP(x) fournit l'exponentielle d'un nombre x, résultat en format virgule flottante (base e).
'FBED	INKEY$	16	Prélève dans le registre A le contenu du tampon clavier à l'adresse '4280. Si le tampon clavier contient 0, branchement à la routine 'F883, on place 0 en 'D0 et retour écran. Dans le cas contraire, une touche a été enfoncée, il faut alors remettre à 0 le tampon clavier et réinitialiser la ligne correspondant à la touche enfoncée '4230+... Le code prélevé est placé en 'CD puis traduit en caractère par la routine 'EE91 et affiché à l'écran.
'F335	INT	01	Le nombre dont on veut prélever la partie entière est stocké en virgule flottante dans la zone intermédiaire réservée aux nombres : 'C9 à 'CD. Un test est effectué sur le contenu de 'C9. S'il est supérieur à 'A0, alors retour. Sinon exécution de la routine 'F30B déjà utilisée lors de la traduction d'un nombre entier sur deux octets (appelé alors par routine 'EBC7 et 'EF4C). Le nombre est alors transformé en entier, mais toujours stocké dans la même zone : 'C9 à CD (en fait en 'CA:'CB).
'EEAD	LEFT$	11	Renvoie les n premiers caractères d'une chaîne. Utilisé sous la forme LEFT$ (chaîne, n).
'EE82	LEN	0C	Renvoie le nombre de caractères d'une chaîne, sous la forme LEN (chaîne).
'F0B9	LOG	06	Calcul du logarithme népérien d'un nombre. Stockage en virgule flottante dans la zone intermédiaire 'C9 à 'CD par routine 'ECE2 (déjà utilisée par INT et LEN).

Adresse hexa de la routine	Mot-clé	Position dans la table (hexa)	Rôle
'ECDB	MEM	17	Calcule l'espace disponible en mémoire en effectuant la différence entre les pointeurs de sommet de pile opérationnelle et le pointeur de fin de la table des variables.
'EECF	MID$	13	MID$ (chaîne, x, y) extrait de la chaîne la partie commençant en caractères de position x sur une longueur y de caractères.
'EF5F	PEEK	0B	Prélève l'argument entier et le traduit en entier sur deux octets ('EF4F) dans le registre IX. Prélève alors le contenu de l'octet mémoire correspondant qui est placé dans le registre B. Puis un branchement à 'ECE2 est effectué. '84 est mis à 0 tandis que le contenu à retourner est placé en 'CA:'CB puis transformé en simple précision par la routine 'F2C3. Cette routine 'ECE2 peut servir à retourner à la fois des entiers sur un octet (dans ce cas on se branche en 'ECE2), ou sur deux octets (on se branche alors en 'ECE3). En effet, lors du branchement, la valeur à retourner est stockée dans le double accumulateur A:B et l'instruction en 'ECE2 a pour rôle de mettre 0 dans A.
'FB9C	POINT	14	Vérifie si un point graphique placé en (x, y) est allumé ou éteint en procédant à des décalages logiques à droite du code prélevé dans la RAM vidéo. Cette fonction renvoie 0 si le point est éteint. 1 à 8 selon la couleur s'il est allumé, – 1 s'il est occupé par un caractère. La valeur renvoyée est retournée par la routine 'ECE3 (donc sur un octet) déjà vue ci-dessus.
'EEC8	RIGHT$	12	RIGHT$ (chaîne, x) extrait les x derniers caractères de la chaîne.
'F62A	RND	04	Création d'un nombre aléatoire compris entre 0 et 1 ou 1 et n selon la valeur de l'argument. Si la valeur de l'argument est 0, le nombre généré est compris entre 0 et 1. Si la valeur de l'argument est strictement positive avec éventuellement un partie décimale, le nombre généré est un entier positif donc la valeur est comprise entre 1 et la partie entière de l'argument.

Adresse hexa de la routine	Mot-clé	Position dans la table (hexa)	Rôle
'F2BA	SGN	0	L'argument étant stocké en virgule flottante dans la zone 'C9 à 'CD, un test est effectué sur 'C9 (si 'C9 est à 0, alors le nombre est nul), puis sur 'CA (voir format de stockage des nombres). Cette fonction renvoie 0 si le nombre est nul, – 1 s'il est négatif et 1 s'il est positif. Le résultat dans B est 1 si le nombre est positif et 'FF s'il est négatif.
'F68C	SIN	08	SIN(x) rend le sinus d'un angle x exprimé en radian, résultat en virgule flottante dans la zone 'C9 à 'CD.
'F54D	SQR	05	SQR(x) rend la racine carrée d'un nombre positif (un test est d'abord effectué sur le signe de x) en virgule flottante.
'ECED	STR$	0D	Change le type d'une expression numérique qu'elle transforme en chaîne. Cette chaîne est placée dans le buffer de sortie (à partir de l'adresse '4334 et le registre d'index est positionné sur la valeur '4333).
'F6D2	TAN	0A	Calcule la tangente d'un angle mesuré en radians.
'4215	USR	03	Branchement vers une sous-routine en langage machine en passant un paramètre. Cette fonction peut restituer la valeur qu'elle calcule (voir chapitre sur les instructions cachées de BASIC).
'EF1C	VAL	0E	VAL (chaîne) change le type d'une expression chaîne et en fait un nombre si cette chaîne est composée de chiffres. Cette fonction, inverse de STR$, fournit un résultat dans la zone réservée aux nombres.
'FC11	VAPRTR	15	Fournit l'adresse de la variable qui constitue son argument s'il s'agit d'une variable numérique ou de son descripteur s'il s'agit d'une variable alphanumérique. L'argument doit impérativement être le nom d'une variable utilisée dans le programme ou d'un élément de tableau. Si aucune valeur n'a encore été affectée à la variable, VARPTR rend la valeur 0.

TABLE DE CRÉATION DES ATTRIBUTS DU BASIC

Chaque attribut est calculé suivant la formule

attribut = '80 + index de la position du mot-clé dans la table

(la table commence en 'E045)

Adresse hexa	Adresse Décimale	Contenu Hexa	Position	Mot-clé	Attribut Hexa	Attribut ~~Hexa~~ Décimal
E045-E047	57413-57415	46.4F.D2	0	FOR	80	128
E048-E04B	57416-57419	47.4F.54.CF	1	GOTO	81	129
E04C-E050	57420-57424	47.4F.53.55.C2	2	GOSUB	82	130
E051-E053	57425-57427	52.45.CD	3	REM	83	131
E054-E055	57428-57429	49.C6	4	IF	84	132
E056-E059	57430-57433	44.41.54.C1	5	DATA	85	133
E05A-E05E	57434-57438	50.52.49.4E.D4	6	PRINT	86	134
E05F-E060	57439-57440	4F.CE	7	ON	87	135
E061-E065	57441-57445	49.4E.50.55.D4	8	INPUT	88	136
E066-E068	57446-57448	45.4E.C4	9	END	89	137
E069-E06C	57449-57452	4E.45.58.D4	0A	NEXT	8A	138
E06D-E06F	57453-57455	44.49.CD	0B	DIM	8B	139
E070-E073	57456-57459	52.45.41.C4	0C	READ	8C	140
E074-E076	57460-57462	4C.45.D4	0D	LET	8D	141
E077-E079	57463-57465	52.55.CE	0E	RUN	8E	142
E07A-E080	57466-57472	52.45.53.54.4F.52.C5	0F	RESTORE	8F	143
E081-E086	57473-57478	52.45.54.55.52.CE	10	RETURN	90	144
E087-E08A	57479-57482	53.54.4F.D0	11	STOP	91	145
E08B-E08E	57483-57486	50.4F.4B.C5	12	POKE	92	146
E08F-E092	57487-57490	43.4F.4E.D4	13	CONT	93	147
E093-E096	57491-57494	4C.49.53.D4	14	LIST	94	148
E097-E09B	57495-57499	43.4C.45.41.D2	15	CLEAR	95	149
E09C-E09E	57500-57502	4E.45.D7	16	NEW	96	150
E09F-E0A3	57503-57507	43.4C.4F.41.C4	17	CLOAD	97	151
E0A4-E0A8	57508-57512	43.53.41.56.C5	18	CSAVE	98	152
E0A9-A0AD	57513-57517	4C.4C.49.53.D4	19	LLIST	99	153
E0AE-E0B3	57518-57523	4C.50.52.49.4E.D4	1A	LPRINT	9A	154
E0B4-E0B6	57524-57526	53.45.D4	1B	SET	9B	155
E0B7-E0BB	57527-57531	52.45.53.45.D4	1C	RESET	9C	156
E0BC-E0BE	57532-57534	43.4C.D3	1D	CLS	9D	157
E0BF-E0C3	57535-57539	53.4F.55.4E.C4	1E	SOUND	9E	158
E0C4-E0C7	57540-57543	45.58.45.C3	1F	EXEC	9F	159
E0C8-E0CC	57544-57548	53.4B.49.50.C6	20	SKIPF	A0	160
E0CD-E0D0	57549-57552	54.41.42.A8	21	TAB(	A1	161
E0D1-E0D2	57553-57554	54.CF	22	TO	A2	162
E0D3-E0D6	57555-57558	54.48.45.CE	23	THEN	A3	163
E0D7-E0D9	57559-59561	4E.4F.D4	24	NOT	A4	164
E0DA-E0DD	59562-59565	53.54.45.D0	25	STEP	A5	165
E0DE-E0E0	59566-59568	4F.46.C6	26	OFF	A6	166
E0E1	59569	AB	27	+	A7	167
E0E2	59570	AD	28	–	A8	168

Adresse hexa	Adresse Décimale	Contenu Hexa	Position	Mot-clé	Attribut Hexa	Attribut Hexa Décimal
E0E3	59571	AA	29	*	A9	169
E0E4	59572	AF	2A	/	AA	170
E0E5	59573	DE	2B	↑	AB	171
E0E6-E0E8	59574-59576	41.4E.C4	2C	AND	AC	172
E0E9-E0EA	59577-59578	4F.D2	2D	OR	AD	173
E0EB	59579	BE	2E	>	AE	174
E0EC	59580	BD	2F	=	AF	175
E0ED	59581	BC	30	<	B0	176
E0EE-E0F0	59582-59584	53.47.CE	31	SGN	B1	177
E0F1-E0F3	59585-59587	49.4E.D4	32	INT	B2	178
E0F4-E0F6	59588-59590	41.42.D3	33	ABS	B3	179
E0F7-E0F9	59591-59593	55.53.D2	34	USR	B4	180
E0FA-E0FC	59594-59596	52.4E.C4	35	RND	B5	181
E0FD-E0FF	59597-59599	53.51.D2	36	SQR	B6	182
E100-E102	59600-59602	4C.4F.C7	37	LOG	B7	183
E103-E105	59603-59605	45.58.D0	38	EXP	B8	184
E106-E108	59606-59608	53.49.CE	39	SIN	B9	185
E109-E10B	59609-59611	43.4F.D3	3A	COS	BA	186
E10C-E10E	59612-59614	54.41.CE	3B	TAN	BB	187
E10F-E112	59615-59618	50.45.45.CB	3C	PEEK	BC	188
E113-E115	59619-59621	4C.45.CE	3D	LEN	BD	189
E116-E119	59622-59625	53.54.52.A4	3E	STR$	BE	190
E11A-E11C	59626-59628	56.41.CC	3F	VAL	BF	191
E11D-E11F	59629-59631	41.53.C3	40	ASC	C0	192
E120-E123	59632-59635	43.48.52.A4	41	CHR$	C1	193
E124-E128	59636-59640	4C.45.46.54.A4	42	LEFT$	C2	194
E129-E12E	59641-59646	52.49.47.48.54.A4	43	RIGHT$	C3	195
E12F-E132	59647-59650	4D.49.44.A4	44	MID$	C4	196
E133-E137	59651-59655	50.4F.49.4E.D4	45	POINT	C5	197
E138-E13D	59656-59661	56.41.52.50.54.D2	46	VARPTR	C6	198
E13E-E143	59662-59667	49.4E.4B.45.59.A4	47	INKEY$	C7	199
E144-E146	59668-59670	4D.45.CD	48	MEM	C8	200
E147	59671	00 (marqueur fin de liste)	49	—	C9	211

Chaque mot est repérable, dans la table, par le code ASCII de chacune des lettres qui le composent, à l'exception de la dernière lettre du mot pour laquelle on trouve le code ASCII + 128 (en décimal) ou code ASCII + '80 (en hexadécimal), ce qui revient à mettre le bit 7 du dernier octet à 1.

Exemple

ASC → 41.53.C3 dans la table

Code ASCII : **41** . **53** . **43**

inchangé — C3-80 en hexa

Les calculs étant effectués selon une priorité décroissante des opérateurs, je pense qu'il peut être utile de terminer par une table triée par priorité décroissante.

Opérateur	Priorité hexa	Priorité décimale	Vecteur de branchi hexa	Vecteur de branchi décimal
I	7F	127	F556	62806
*	7B	123	F0F1	61861
/	7B	123	F1C8	61896
+	79	121	EF80	61312
–	79	121	EF75	61301
AND	50	80	EA8E	60046
OR	46	70	EA8D	60045

Lorsque deux opérateurs ont le même rang de priorité (* et / ou + et –, par exemple), les calculs sont alors effectués de gauche à droite dans l'expression.

TABLE DE PRIORITÉ DES OPÉRATEURS

Lors des calculs mathématiques, logiques, une priorité est affectée à chaque opérateur. Les calculs sont alors effectués par ordre de priorité décroissante. Les rangs de priorité entre opérateurs sont définis dans les deux tables suivantes. Ils sont utilisés pendant la phase d'analyse des instructions BASIC et permettent le stockage dans la pile opérationnelle des résultats intermédiaires. Dans la table, débutant à l'adresse 'E030 de la ROM, on trouve pour chaque opérateur sa priorité sur un octet, suivie du vecteur de branchement à la routine d'exécution.

CLASSEMENT PAR ADRESSE

Adresse hexa	Adresse décimale	Valeur de priorité hexa	Valeur de priorité déc.	Opérateur	Vecteur hexa	Vecteur déc.
E030	57392	79	121	+	EF80	61312
E033	57395	79	121	–	EF75	61301
E036	57398	7B	123	*	F0F1	61681
E039	57401	7B	123	/	F1C8	61896
E03C	57404	7F	127	↑	F556	62806
E03F	57407	50	80	AND	EA8E	60046
E042	57410	46	70	OR	EA8D	60045

ORDRE DE PRIORITÉ DES OPÉRATEURS RELATIONNELS

<=	06
<>	05
>=	04
<	03
=	02
>	01

ROUTINES ASSOCIÉES AUX ATTRIBUTS

Chaque attribut se voit associer une adresse de routine de la façon suivante. Il y a trois types d'attributs :

- ceux associés à des mots-clés d'instructions ;
- ceux associés à des opérateurs ;
- ceux associés à des fonctions.

Pour les mots-clés d'instructions, la table contenant les adresses des routines associées débute à l'adresse 'E148. Pour prélever l'adresse de la routine correspondant à un attribut compris en '80 et 'A0 (128 et 160 en décimal), on procède de la façon suivante (routine située à l'adresse $E54F). à $E565

```
CMP   A,'A0     } (A contient l'attribut du mot-clé). Si code >160, ce n'est pas un
BHI   ,'E527    } mot-clé d'instruction, donc erreur de syntaxe.
ASL   A           Décalage arithmétique à gauche de A qui donne le nombre d'oc-
TAB               tets de déplacement par rapport à 'E148, adresse de début, pour
LDX   ,'E148      obtenir l'adresse de la routine.
ABX
LDX   ,('00,X)    chargement dans IX de l'adresse de la routine.
```

Pour les attributs associés aux opérateurs, la table débute en : E031 hexa, soit 57393 en décimal. Quant aux attributs associés à des fonctions, la table des adresses commence en : E000 hexa, soit 57344 en décimal.

CLASSEMENT PAR NUMÉRO D'ATTRIBUT

Mot-clé	Attribut		Adresse table		Adresse routines	
	Hexa	Décimal	Hexa	Décimal	Hexa	Décimal
FOR	80	128	E148	57672	E4C4	58564
GOTO	81	129	E14A	57674	E61A	58906
GOSUB	82	130	E14C	57676	E604	58884
REM	83	131	E14E	57678	E685	59013
IF	84	132	E150	57680	E672	58994
DATA	85	133	E152	57682	E651	58961
PRINT	86	134	E154	57684	E71F	59167
ON	87	135	E156	57686	E692	59026
INPUT	88	136	E158	57688	E7DE	59358
END	89	137	E15A	57690	E57F	58751
NEXT	8A	138	E15C	57692	E8BB	59596
DIM	8B	139	E15E	57694	EB12	60178
READ	8C	140	E160	57696	E80E	59406
LET	8D	141	E162	57698	E6D3	59091
RUN	8E	142	E164	57700	E5FA	58874
RESTORE	8F	143	E166	57702	E560	58720
RETURN	90	144	E168	57704	E631	58929
STOP	91	145	E16A	57706	E57E	58750
POKE	92	146	E16C	57708	EF66	61286
CONT	93	147	E16E	57710	E5A6	58790
LIST	94	148	E170	57712	E40D	58381
CLEAR	95	149	E172	57714	E5B8	58808
NEW	96	150	E174	57716	E3CD	58317
CLOAD	97	151	E176	57718	FD5C	64860
CSAVE	98	152	E178	57720	FC3B	64571
LLIST	99	153	E17A	57722	E40B	58379
LPRINT	9A	154	E17C	57724	E71C	59164
SET	9B	155	E17E	57726	FB25	64293
RESET	9C	156	E180	57728	FB55	64341
CLS	9D	157	E182	57730	FBBF	64447
SOUND	9E	158	E184	57732	FFA5	65445
EXEC	9F	159	E186	57734	FC04	64516
SKIPF	A0	160	E188	57736	FE2F	65071
TAB(	A1	161			E78C	59276
TO	A2	162			E4E1	58593
THEN	A3	163			E67C	59004
NOT	A4	164			EA13	59223
STEP	A5	165			E507	58631
OFF	A6	166				
+	A7	167	E031	57393	EF80	61312

Mot-clé	Attribut		Adresse table		Adresse routine	
	Hexa	Décimal	Hexa	Décimal	Hexa	Décimal
–	A8	168	E034	57396	EF75	61301
*	A9	169	E037	57399	F0F1	61681
/	AA	170	E03A	57402	F1C8	61896
↑	AB	171	E03D	57405	F556	62808
AND	AC	172	E040	57408	EA8E	60046
OR	AD	173	E043	57411	EA8D	60045
>	AE	174				
=	AF	175			E6D8	59096
<	B0	176				
SGN	B1	177	E000	57344	F2BA	62138
INT	B2	178	E002	57346	F335	62261
ABS	B3	179	E004	57348	F2D5	62165
USR	B4	180	E006	57350	4215	16217
RND	B5	181	E008	57352	F62A	63018
SQR	B6	182	E00A	57354	F54D	62797
LOG	B7	183	E00C	57356	F0B9	61625
EXP	B8	184	E00E	57358	F5C9	62921
SIN	B9	185	E010	57360	F68C	63116
COS	BA	186	E012	57362	F686	63110
TAN	BB	187	E014	57364	F6D2	63186
PEEK	BC	188	E016	57366	EF5F	61279
LEN	BD	189	E018	57368	EE82	61058
STR$	BE	190	E01A	57370	ECED	60653
VAL	BF	191	E01C	57372	EF1C	61212
ASC	C0	192	E01E	57374	EEA2	61090
CHR$	C1	193	E020	57376	EE8E	61070
LEFT$	C2	194	E022	57378	EEAD	61101
RIGHT$	C3	195	E024	57380	EEC8	61128
MID$	C4	196	E026	57382	EECF	61135
POINT	C5	197	E028	57384	FB9C	64412
VARPTR	C6	198	E02A	57386	FC11	64529
INKEY$	C7	199	E02C	57388	FBED	64493
MEM	C8	200	E02E	57390	ECDB	60635

LIMITES ET CONSOMMATION DE MÉMOIRE

LIMITES

Variables numériques : de −1.70141183E+38 à 1.70141183E+38.
Chaînes alphanumériques : 255 caractères maximum.
Numeros de lignes de programme : de 0 à 63999 inclus.
Longueur des lignes de programme : 4 lignes de 32 caractères, soit 128 caractères.

CONSOMMATION DE MÉMOIRE

Chaque ligne de programme nécessite au moins 6 octets.
- Pointeur ligne suivante : 2 octets ;
- numéro de ligne : 2 octets ;
- fin de ligne : 1 octet (toujours à 0).

De plus, chaque caractère de la ligne de programme occupe un octet à l'exception des mots réservés qui n'occupent chacun qu'un seul octet.

A L'EXÉCUTION

- Variable numérique : 7 octets (nom : 2 octets, valeur : 5 octets).
- Variable alphanumérique : 6 octets (nom : 2 octets, longueur : 2 octets. Pointeur vers la valeur : 2 octets + 1 octet par caractère).
- Tableau : nom : 2 octets, taille totale en nombre d'octets : 2 octets, nombre de dimensions : 1 octet, taille : 2 octets par dimension. Chaque élément : 5 octets.
- Boucle FOR-NEXT : chaque boucle active nécessite 18 octets.
- GOSUB : chaque GOSUB actif nécessite 7 octets.

PROGRAMMES D'APPLICATIONS

- DESASSEMBLEUR
- DUMP DE LA MEMOIRE
- ROUTINES DE SAUVEGARDE D'UNE ZONE MEMOIRE SUR CASSETTE
- EXEMPLE D'UTILISATION DE L'ECRAN EN MODE GRAPHIQUE
- JEU « SIMONE »
- JEU DES ANIMAUX
- GESTION DE COMPTE- CHEQUES
- ANNEXE

DESASSEMBLEUR

Ce désassembleur 6803 vous permettra de visualiser les séquences de programme, écrites en code machine, en langage assembleur 6803. Vous trouverez un exemple, en fin de listing, d'une exécution.

L'utilisation en est très simple :

1. Le programme vous demande si vous avez une imprimante.
2. Les variables sont initialisées (message "UN INSTANT SVP").
3. Puis le programme vous demande l'adresse hexa (sur quatre chiffres) à partir de laquelle vous souhaitez désassembler le contenu de la mémoire.

Pour arrêter l'exécution, utiliser la touche "BREAK".

Description de la structure du programme

lignes 10 à 140	: initialisation du programme.
lignes 200 à 250	: désassemblage de la ligne d'instruction.
lignes 300 à 310	: cas d'un adressage implicite.
lignes 320 à 340	: cas d'un adressage relatif (branchements).
lignes 350 à 370	: cas d'un adressage immédiat étendu (double accumulateur).
lignes 400 à 420	: cas d'un adressage immédiat.
lignes 500 à 510	: cas d'un adressage direct.
lignes 600 à 620	: cas d'un adressage indexé.
lignes 700 à 720	: cas d'un adressage étendu.
lignes 900 à 920	: décodage de la première partie de la ligne d'instruction.
lignes 1000 à 1030	: transformation d'un octet en hexadécimal.
lignes 1200 à 1230	: transformation de trois octets.
lignes 1300 à 1330	: transformation hexadécimal en décimal.
lignes 1400 à 1430	: transformation de quatre octets.
lignes 2000 à 2030	: impression page titre
lignes 9000 à 10340	: données.

```
10 CLEAR 10000
20 CLS  GOSUB 2600
30 PRINT@128,"AVEZ-VOUS UNE IMPR
IMANTE (O/N)";:Z2$=INPUT$(1):IF Z2$
="" THEN 40
50 IF Z2$="O" THEN POKE17756,154
55 IF Z2$="N" THEN POKE17756,134
70 $=1
80 B$=""
90 H$="0123456789ABCDEF"
100 DIM IN$(255),T(255),L$(8):RE
STORE:FORI=1TO 8:READL$(I):NEXTI
105 CLS3:GOSUB 2000:PRINT@325,"U
N INSTANT S.V.P.....";
110 FORI=0TO255
120 READ A,A$,IN$(A)
130 FOR K=1 TO 8:IF A$=L$(K) THE
N T(A)=K:GOTO 140
135 NEXTK
140 NEXTI
200 PRINT@ 325,"ADRESSE HEXA ";:
INPUT A$:GOSUB 1300
210 GOSUB 1200
220 Z$="":Z1$="":Z2$="":Z$=A$+"
"
230 GOSUB 1000
240 ON T(I) GOSUB 300,300,800,50
0,600,700,320,350
250 LPRINT Z$;TAB(17);Z1$;TAB(21
);Z2$:A=A+1
260 GOTO 210
300 Z$=Z$+I$+" ":Z1$=IN$(I)
310 RETURN
320 Z$=Z$+I$+" ":A=A+1:I1=I:GOSU
B1500
330 Z$=Z$+I$:Z1$=IN$(I1):Z2$=",'
":B=A:IF I>127 THEN I=I-256
340 A=A+1+I:GOSUB1200:Z2$=Z2$+A$
:A=B:RETURN
350 Z$=Z$+I$+" ":GOSUB900
360 Z1$=IN$(I1):Z2$=",'"+A$
370 RETURN
400 Z$=Z$+I$+" ":A=A+1:I1=I:GOSU
B1500
410 Z$=Z$+I$:Z1$=IN$(I1):Z2$="
"+I$
420 RETURN
500 Z$=Z$+I$+" ":A=A+1:I1=I:GOSU
B 1500
510 Z$=Z$+I$:Z1$=IN$(I1):Z2$=",(
'"+I$+")":RETURN
600 Z$=Z$+I$+" ":A=A+1:I1=I:GOSU
B1500:Z$=Z$+I$
610 Z1$=IN$(I1):Z2$=",('"+I$+",X
)"
620 RETURN
700 Z$=Z$+I$+" ":GOSUB 900
710 Z1$=IN$(I1):Z2$=",('"+A$+")"
720 RETURN
800 GOSUB 400:RETURN
900 V=0:I1=I:A=A+1:GOSUB1500:Z$=
Z$+I$+" ":V=I
910 A=A+1:GOSUB 1500:Z$=Z$+I$:V=
V*256+I:B=A:A=V:GOSUB 1200:A=B
920 RETURN
1000 I=PEEK(A)
1010 G1=INT(I/16):G2=I-(INT(I/16
)*16)
1020 I$=MID$(H$,G1+1,1)+MID$(H$,
G2+1,1)
1030 RETURN
1200 A$="":A1=A:FORJ=3 TO 0 STEP
-1
1220 A$=A$+MID$(H$,INT(A1/16^J)+
1,1):A1=A1-(INT(A1/16^J)*16^J)
1230 NEXT J:RETURN
1300 A=0:FORJ=1 TO 4
1310 FOR K=1 TO  16:IF MID$(A$,J
,1)=MID$(H$,K,1) THEN GOTO1330
1320 NEXT K
1330 A=A*16-1+K:NEXTJ:RETURN
1400 I=0:FORJ=1 TO 2
1410 FORK=1 TO 16:IF MID$(I$,J,1
)=MID$(H$,K,1) THEN K=K-1:GOTO 1
430
1420 NEXT K
```

```
1430 I=I+16-1:K NEXTJ RETURN
1500 GOSUB 1700 GOSUB 1000 RETUR
N
2000 PRINT@39,"DESASSEMBLEUR 680
3";
2010 PRINT@71,"-----------------
-";
2020 PRINT@454,"(C) A. BONNEAUD
1984";
2030 RETURN
9000 DATA INH,SAN,IMM,DIR,IND,EX
T,REL,IME
10000 DATA 27,INH,ABA,137,IMM,AD
C A,153,DIR,ADC A,185,EXT,ADC A,
169,IND,ADC A,201,IMM,ADC B
10010 DATA 217,DIR,ADC B,249,EXT
,ADC B,233,IND,ADC B,139,IMM,ADD
 A
10020 DATA 155,DIR,ADD A,187,EXT
,ADD A,171,IND,ADD B,203, IMM, A
D B
10030 DATA 219,DIR,ADD B,251,EXT
,ADD B,235,IND,ADD B,132,IMM,AND
 A
10040 DATA 148,DIR,AND A,180,EXT
,AND A,164,IND,AND A,196,IMM,AND
 B
10050 DATA 212,DIR,AND B,244,EXT
,AND B,228,IND,AND B,72,SAN,ASL
 A
10060 DATA 88,SAN,ASL B,120,EXT,
ASL,104,IND,ASL ,71,SAN,ASR A
10070 DATA 87,SAN,ASR B,119,EXT,
ASR,103,IND,ASR ,36,REL,BCC,37,R
EL,BCS,39,REL,BEQ,44,REL,BGE,46,
REL,BGT
10080 DATA 34,REL,BHI,133,IMM,BI
T A,149,DIR,BIT A,181,EXT,BIT A,
165,IND,BIT A,197,IMM,BIT B,213,
DIR,BIT B,245,EXT,BIT B
10090 DATA 229,IND,BIT B,47,REL,
BLE,35,REL,BLS,45,REL,BLT,43,REL
,BMI,38,REL,BNE,42,REL,BPL,32,RE
L,BRA,141,REL,BSR,40,REL,BVC
10100 DATA 41,REL,BVS,17,INH,CBA
,12,INH,CLC,14,INH,CLI,79,SAN,CL
R A,95,SAN,CLR B,127,EXT,CLR,111
,IND,CLR
10110 DATA 10,INH,CLV,129,IMM,CM
P A,145,DIR,CMP A,177,EXT,CMP A,
161,IND,CMP A,193,IMM,CMP B,209,
DIR,CMP B,241,EXT,CMP B
10120 DATA 225,IND,CMP B,67,SAN,
COM A,83,SAN,COM B,115,EXT,COM,9
9,IND,COM,140,IME,CPX,156,DIR,CP
X,188,EXT,CPX
10130 DATA 172,IND,CPX,25,INH,DA
A,74,SAN,DEC A,90,SAN,DEC B,122,
EXT,DEC,106,IND,DEC,52,INH,DES,9
,INH,DEX
10140 DATA 136,IMM,EOR A,152,DIR
,EOR A,184,EXT,EOR A,168,IND,EOR
 A,200,IMM,EOR B,216,DIR,EOR B,2
48,EXT,EOR B,232,IND,EOR B
10150 DATA 76,SAN,INC A,92,SAN,I
NC B,124,EXT,INC,108,IND,INC,49,
INH,INS,8,INH,INX,126,EXT,JMP,11
0,IND,JMP
10160 DATA 189,EXT,JSR,173,IND,J
SR,134,IMM,LDA A,150,DIR,LDA A,1
82,EXT,LDA A,166,IND,LDA A,198,I
MM,LDA B,214,DIR,LDA B
10170 DATA 246,EXT,LDA B,230,IND
,LDA B,142,IME,LDS,158,DIR,LDS,1
90,EXT,LDS,174,IND,LDS,206,IME,L
DX,222,DIR,LDX
10180 DATA 254,EXT,LDX,238,IND,L
DX,68,SAN,LSR A,84,SAN,LSR B,116
,EXT,LSR,100,IND,LSR,4,SAN,NEG
A,80,SAN,NEG B
10190 DATA112,EXT,NEG,96,IND,NEG
,1,INH,NOP,138,IMM,ORA A,154,DIR
,ORA A,186,EXT,ORA A,170,IND,ORA
 A,202,IMM,ORA B
10200 DATA 218,DIR,ORA B,250,EXT
,ORA B,234,IND,ORA B,54,SAN,PSH
```

```
N,55,SAN,PSH B,56,SAN,PUL A,51,S
AN,PUL B,73,SAN,ROL A
10210 DATA 89,SAN,ROL B,121,EXT,
ROL,105,IND,ROL,70,SAN,ROR A,86,
SAN,ROR B,118,EXT,ROR,102,IND,RO
R,59,INH,RTI
10220 DATA 57,INH,RTS,16,INH,SBA
,130,IMM,SBC A,146,DIR,SBC A,178
,EXT,SBC A,162,IND,SBC A,194,IMM
,SBC B,210,DIR,SBC B
10230 DATA 242,EXT,SBC B,226,IND
,SBC B,13,INH,SEC,15,INH,SEI,11,
INH,SEV,151,DIR,STA A,183,EXT,ST
A A,167,IND,STA A
10240 DATA 215,DIR,STA B,247,EXT
,STA B,231,IND,STA B,159,DIR,STS
,191,EXT,STS,175,IND,STS,223,DIR
,STX,255,EXT,STX
10250 DATA 239,IND,STX,128,IMM,S
UB A,144,DIR,SUB A,176,EXT,SUB A
,160,IND,SUB A,192,IMM,SUB B,208
,DIR,SUB B,240,EXT,SUB B
10260 DATA 224,IND,SUB B,63,INH,
SWI,22,INH,TAB,6,INH,TAP,23,INH,
TBA,7,INH,TPA,77,SAN,TST A,93,SA
N,TST B
10270 DATA 125,EXT,TST,109,IND,T
ST,48,INH,TSX,53,INH,TXS
10280 DATA 62,INH,WAI,0,INH,???,
2,INH,???,3,INH,???,4,INH,LSRD,5
,INH,ASLD,10,INH,???,19,INH,???,
20,INH,???,21,INH,???
10290 DATA 24,INH,???,26,INH,???
,28,INH,???,29,INH,???,30,INH,??
?,31,INH,???,33,REL,BRN,56,INH,P
UL X,58,INH,ABX,60,INH,PSH X
10300 DATA 61,INH,MUL,65,INH,???
,66,INH,???,69,INH,???,75,INH,??
?,78,INH,???,81,INH,???,82,INH,?
??,85,INH,???,91,INH,???
10310 DATA 94,INH,???,97,INH,???
,98,INH,???,101,INH,???,107,INH,
???,115,INH,???,114,INH,???,117,
INH,???,123,INH,???
10315 DATA 131,IME,SUBD,135,IMM,
???
10320 DATA 143,INH,???,147,DIR,S
UBD,157,DIR,JSR,163,IND,SUBD,179
,EXT,SUBD,195,IME,ADDD,199,INH,?
??,204,IME,LDD,205,INH,???
10330 DATA 207,INH,???,211,DIR,A
DDD,220,DIR,LDD,221,DIR,STD,22
7,IND,ADDD,236,IND,LDD,237,IND,
STD,243,EXT,ADDD
10340 DATA 252,EXT,LDD,253,EXT,
STD
```

EXEMPLE DE DESASSEMBLAGE

```
FAF1 5A          DEC B
FAF2 27 03       BEQ ,'FAF7
FAF4 86 08       LDA A,'08
FAF6 BD F9 C6    JSR ,('F9C6)
FAF9 20 F6       BRA ,'FAF1
FAFB 81 03       CMP A,'03
FAFD 0D          SEC
FAFE 27 05       BEQ ,'FB05
FB00 81 0D       CMP A,'0D
FB02 26 0E       BNE ,'FB12
FB04 4F          CLR A
FB05 07          TPA
FB06 36          PSH A
FB07 BD E7 66    JSR ,('E766)
FB0A 6F'0A       CLR ,('00,X)
FB0C CE 42 B1    LDX ,'42B1
FB0F 32          PUL A
FB10 06          TAP
FB11 39          RTS
FB12 81 20       CMP A,'20
FB14 25 99       BCS ,'FAAF
FB16 8D 02       BSR ,'FB1A
FB18 20 95       BRA ,'FAAF
FB1A C1 30       CMP B,'30
FB1C 24 F3       BCC ,'FB11
FB1E A7 00       STA A,('00,X)
FB20 08          INX
FB21 5C          INC B
FB22 7E F9 C6    JMP ,('F9C6)
FB25 BD 43       BSR ,'FB6A
FB27 3C          PSH X
FB28 BD EF 47    JSR ,('EF47)
```

DUMP DE LA MEMOIRE

Ce programme de "vidage" de la mémoire permet d'afficher à l'écran les contenus hexadécimaux de tous les octets de la mémoire entre les deux adresses constituant les bornes. On affiche ainsi des lignes de 16 octets, précédées de l'adresse du premier octet de la ligne et suivies de la traduction en code ASCII du contenu de chaque octet ou d'un point lorqu'aucun code ASCII ne correspond à ce contenu.

Ce DUMP permet un choix entre l'écran et l'imprimante. Toutefois, une ligne représentant 72 caractères, il est nécessaire, pour pouvoir l'utiliser au mieux de disposer d'une imprimante 80 colonnes. Cependant ce programme fonctionnera sur la petite imprimante thermique PC10 de Tandy sur 32 colonnes mais chaque ligne de DUM nécessitera trois lignes d'imprimante.

Fonctionnement

Le programme vous demande d'abord de sélectionner l'écran ou l'imprimante, puis vous réclame les bornes entre lesquelles vous souhaitez faire le dump. Ces bornes doivent être des adresses exprimées en décimal, dans l'ordre croissant.
Si vous avez choisi l'écran, la ligne 5040 place aux adresses 17972, 18032, 18106 le code du "PRINT" (PT = 133) ; dans le cas contraire, elle place à ces adresses le code du "LPRINT" (PT = 154). Il s'agit bien sûr des ordres d'impression qui apparaissent dans les lignes 5150, 5180 et 5200 du programme. Si vous ne respectez pas scrupuleusement le format du listing donné ci-après, lors de sa saisie, vous devrez recalculer ces adresses.

Structure du programme

lignes 0 à 5022	: initialisation.
lignes 5024 à 5040	: choix du périphérique de sortie (écran ou imprimante).
lignes 5045 à 5060	: définitions des adresses de sortie (écran ou imprimante).
ligne 5065	: appel de la routine de "DUMP".
lignes 5070 à 5150	: détermination de l'adresse hexadécimale du premier octet de la ligne.
lignes 5160 à 5220	: affichage de la ligne de "Dump".
lignes 5260 à 5270	: y a-t-il une autre zone à explorer ?

EXEMPLE DE DUMP MEMOIRE

```
0000  FF FF 00 4F FC FD FE FF E0 5C FD FF FF 03 00 FF  . . O. C.
0010  FF 20 00 FF 7F FF FF FF FF FF FF FF FF FF FF FF  . . . B.
0020  FC FD FE FF FC FD FE FF FC FD FE FF FC FD FE FF  .
0030  FC FD FE FF FC FD FE FF FC FD FE FF FC FD FE FF  ..
0040  FC FD FE FF FC FD FE FF FC FD FE FF FC FD FE FF  ..
0050  FC FD FE FF FC FD FE FF FC FD FE FF FC FD FE FF  ..
0060  FC FD FE FF FC FD FE FF FC FD FE FF FC FD FE FF  ..
0070  FC FD FE FF FC FD FE FF FC FD FE FF FC FD FE FF  ..
0080  00 00 4F 00 00 00 00 00 00 4F 4F 47 CB C0 00 00  .      O4NN..
0090  00 4F 5E 43 40 47 9F 47 FA 49 FC 4F 00 4F 0F 4F  .O?CE@.G.HIO.O.O
00A0  E7 4F FE 14 96 15 40 00 00 47 13 00 00 43 43 47  .O..    GR. CEG
00B0  55 48 00 47 D2 47 D3 00 00 00 57 70 6C 00 00 00  .K.G.
00C0  00 47 00 11 00 00 00 42 41 98 00 00 00 CD 4E 00  ..      BA..   N.
```

```
0 CLEAR 1000
1 CLS
10 PT=134
5010 DIM V(2),A$(16):FORI=0 TO9:
A$(I)=CHR$(I+48):NEXT I:FOR I=10
 TO 15:A$(I)=CHR$(I+55):NEXT I
5020 PRINT@36,"D U M P M E M O I
R E";
5022 PRINT@68,"----------------
----";
5024 PRINT@130,"(E)CRAN OU (I)MP
RIMANTE?";
5026 OP$=INKEY$:IF OP$="" THEN 5
026
5030 IF OP$="E" THEN PT=134
5032 IF OP$="I" THEN PT=154
5040 POKE 13941,PT:POKE 14001,PT
:POKE 14077,PT
5045 PRINT @260,"ENTREZ LES BORN
ES"
5050 PRINT:INPUT"DEBUT ";DE:INPU
T"FIN   ";FI:IF DE>FI THEN PRINT
"LES BORNES DOIVENT ETRE ORDONNE
ES" GOTO 5050
5060 DE=DE-(DE/16-INT(DE/16))*16
5065 GOSUB 5100
5067 GOTO 5260
5070 FOR I1=0TO 1
5080 V(I1)=INT((D/16-INT(D/16))*
16)
5090 D=INT(D/16)
5100 IF D<16 THENV(I1+1)=D
5110 NEXTI1:C$=A$(V(1))+A$(V(0))
:RETURN
5120 IF I=0 THEN R$="0000":RETUR
N
5130 R$="":D=I:FORI1=0 TO LOG(I)
/LOG(16)
5140 D=D/16:R$=A$((D-INT(D))*16)
+R$:D=INT(D)
5150 NEXT I1:R$=RIGHT$("0000"+R$
,4):RETURN
5160 FORI=DE TO FI STEP 16:GOSUB
 5120:PRINT R$," "+CHR$(138)+" ";
5170 FOR J=0 TO 15
5180 K=I+J:D=PEEK(K):GOSUB 5070:
PRINT C$;" ";:IF PEEK(K)<32 OR P
EEK(K)>127 THEN D$=D$+".":GOTO52
00
5190 D$=D$+CHR$(PEEK(K))
5200 NEXTJ:PRINT " "+CHR$(138);D
$;CHR$(138)
5210 L$="":D$="":NEXT I
5220 RETURN
5260 PRINT "ENCORE? (<O>UI OU <N
>ON)";
5262 R$=INKEY$:IF R$="" THEN 526
2
5264 IF R$="O" THEN 5030
5270 END
```

ROUTINES DE SAUVEGARDE D'UNE ZONE MEMOIRE SUR CASSETTE

Les routines données ci-après permettent de sauvegarder le contenu d'une zone mémoire sur cassette. Cette partie de mémoire pourra être rechargée en machine par l'instruction CLOADM déjà étudiée dans le chapitre sur les instructions cachées du BASIC.

Vous trouverez deux exemples de routines de sauvegarde :

— l'une utilise le EXEC,

— l'autre est appelée par la fonction USR.

Les exemples choisis permettent la sauvegarde du contenu d'un écran sur cassette, mais vous pouvez sauvegarder toute zone mémoire de la même façon en changeant seulement les adresses. Cependant, pour rappeler la zone sauvée en mémoire centrale, il est nécessaire d'en connaître précisément la longueur nombre d'octets. Pensez donc à la noter au moment de la sauvegarde !

```
10 REM ************************
20 REM *                      *
30 REM *  RESERVATION D'UNE   *
40 REM *  ZONE PROTEGEE EN    *
50 REM *   HAUT DE MEMOIRE    *
60 REM *  POUR Y STOCKER LA   *
70 REM *  ROUTINE DE SAUVE-   *
80 REM *  GARDE SUR CASSETTE  *
90 REM *  ECRITE EN LANGAGE   *
92 REM *        MACHINE.      *
94 REM *                      *
96 REM ************************
100 REM CE PROGRAMME DE RESERVATION DOIT ETRE EXECUTE AVANT DE
CHARGER OU DE SAISIR LES PRO-
GRAMMES SUIVANTS
110 REM
120 REM     -SAUVEGARDE SUR CAS-
             SETTE EN L. M.
130 REM     -JEU DES ANIMAUX
            -GESTION DE COMPTE -
             CHEQUE
140 POKE 16377,150
150 POKE 162,150
160 POKE 158,150
170 EXEC 63340
```

```
10 REM ***************************
20 REM *                         *
30 REM *  VERSION UTILISANT      *
40 REM * L'INSTRUCTION 'EXEC'    *
50 REM * DANS CET EXEMPLE LE     *
60 REM * NOM DU FICHIER EST      *
70 REM *  SUPPOSE PLACE EN       *
80 REM * L'ADRESSE 30000 SOIT    *
90 REM *       EN '7530          *
94 REM *                         *
98 REM ***************************
100 I=36770: REM ADRESSE DE DEBU
T DU CODE MACHINE
110 READ X
120 IF X<0 THEN 160
130 POKE I,X
140 I=I+1
150 GOTO 110
160 END
165 DATA 206
168 REM ADRESSE DE DEBUT ZONE A
SAUVEGARDER
170 DATA 64,0
175 DATA 255,66,111,206
178 REM ADRESSE DE FIN DE ZONE A
SAUVEGARDER
180 DATA 66,0
185 DATA 255,66,113,222,244,255,
143,242,206
188 REM ADRESSE NOM DU FICHIER
(ICI @30000)
190 DATA 117,48
200 DATA 223,244,198,2,189,252,7
9,254,143,242,223,244,57,-1
```

```
*****************************
**                         **
**    LISTING DU PROGRAMME   **
**       DESASSEMBLE.        **
**                         **
*****************************
```

```
8FA2 CE 40 00     LDX ,'4000
8FA5 FF 42 6F     STX ,('426F)
8FA8 CE 42 00     LDX ,'4200
8FAB FF 42 71     STX ,('4271)
8FAE DE F4        LDX ,('F4)
8FB0 FF 8F F2     STX ,('8FF2)
8FB3 CE 75 30     LDX ,'7530
8FB6 DF F4        STX ,('F4)
8FB8 C6 02        LDA B,'02
8FBA BD FC 4F     JSR ,('FC4F)
8FBD FE 8F F2     LDX ,('8FF2)
8FC0 DF F4        STX ,('F4)
8FC2 39           RTS
```

```
1 REM ***************************
2 REM *                         *
3 REM *   SAUVEGARDE D'UN       *
4 REM * FICHIER EN UTILISANT    *
5 REM *  LA ROUTINE VERSION     *
6 REM *        'USR'            *
7 REM *                         *
8 REM ***************************
10 CLEAR 3000
15 PRINT "AVEZ-VOUS UN FICHIER?"
16 X$=INKEY$:IF X$<>"O" AND X$<>
"N" THEN 16
17 IF X$="O" THEN CLS3:GOTO 250
19 REM ON INITIALISE LE PROGRAMM
E EN LANGAGE MACHINE.
20 DIM T$(5)
30 D1=0:D2=0
35 A=0:B=0
40 I=36770
50 READ X
60 IF X<0 THEN 100
70 POKEI,X
80 I=I+1
90 GOTO 50
95 REM SAISIE DES ELEMENTS DU TA
BLEAU T$
100 FOR I=1 TO 5
```

```
110 PRINT "ELEMENT ";I;" DE T$ "
; INPUT T$(I)
120 NEXT I
123 REM SAUVEGARDE DES POINTEURS
, DE LA TABLE DES VARIABLES ET D
U TABLEAU T$
125 POKE 16918,143:POKE16919,162
130 POKE 36771,0:POKE36772,149
140 POKE 36777,0:POKE 36778,162
150 X$=CHR$(34)+"POINTR"+CHR$(34
)+CHR$(0)
160 GOSUB 1000
170 POKE 36771,PEEK(149):POKE 36
772,PEEK(150)
180 POKE 36777,PEEK(153):POKE367
78,PEEK(154)
190 X$=CHR$(34)+"VARIABLE"+CHR$(
34)+CHR$(0)
200 GOSUB 1000
210 POKE 36771,PEEK(155):POKE367
72,PEEK(156)
220 POKE 36777,PEEK(161):POKE367
78,PEEK(162)
230 X$=CHR$(34)+"CHAINE"+CHR$(34
)+CHR$(0)
240 GOSUB 1000
250 PRINT "REMBOBINEZ LA CASSETT
E"
255 PRINT"PUIS METTEZ LE MAGNETO
 EN LECT. ET APPUYEZ SUR UNE TOU
CHE"
260 EXEC 58743
270 CLOADM"POINTR",149-13
275 A=256*PEEK(149)+PEEK(150)
277 B=256*PEEK(153)+PEEK(154)-A
280 CLOADM"VARIABLE",A-B
285 A=256*PEEK(155)+PEEK(156)
287 B=256*PEEK(161)+PEEK(162)-A
290 CLOADM"CHAINE",A-B
300 DATA 206,64,0,255,66,111,206
,66,0,255,66,113,222,244,255,143
,242,189,239,79,223,244,198,2,18
9,252,79,254,143,242,223,244
310 DATA 252,143,242,189,236,227
,57,-1
320 GOTO 2000
1000 D1=VARPTR(X$)+2
1010 D2=256*PEEK(D1)+PEEK(D1+1)
1020 PRINT "MAGNETO ET 'ENTER'"
1030 EXEC 58743
1040 CLS3:PRINT"ENREGISTREMENT D
E ";X$
1050 D1=USR(D2)
1060 CLS3
1070 RETURN
2000 END
```

```
10 REM ************************
20 REM *                      *
30 REM * VERSION UTILISANT    *
40 REM * LA FONCTION 'USR'    *
50 REM *                      *
60 REM ************************
70 I=36770
80 READ X
90 IF X<0 THEN 130
100 POKE I,X
110 I=I+1
120 GOTO 80
130 END
140 DATA 206
150 REM DEBUT DE LA ZONE A SAUVE
GARDER
152 DATA 64,0
160 DATA 255,66,111,206
170 REM FIN DE LA ZONE A SAUVEGA
RDER
172 DATA 66,0
180 DATA 255,66,113,222,244,255,
143,242,189,239,79,223,244,196,2
,189,252,79,254,143,242,223,244
190 DATA 252,143,242,189,236,22
,57,-1
```

```
8FA2 CE 40 00     LDX ,'4000
8FA5 FF 42 6F     STX ,('426F)
8FA8 CE 42 00     LDX ,'4200
8FAB FF 42 71     STX ,('4271)
8FAE DE F4        LDX ,('F4)
8FB0 FF 8F F2     STX ,('8FF2)
8FB3 BD EF 4F     JSR ,('EF4F)
8FB6 DF F4        STX ,('F4)
8FB8 C6 02        LDA B,'02
8FBA BD FC 4F     JSR ,('FC4F)
8FBD FE 8F F2     LDX ,('8FF2)
8FC0 DF F4        STX ,('F4)
8FC2 FC 8F F2     LDAD,('8FF2)
8FC5 BD EC E3     JSR ,('ECE3)
8FC8 39           RTS
```

```
400 REM ************************
410 REM *                      *
420 REM *  EX. DE SAUVEGARDE   *
430 REM * SUR CASSETTE UTILI-  *
440 REM *  SANT LE PROGRAMME   *
450 REM *  EN LANGAGE MACHINE  *
460 REM *  DONNE CI-DESSUS.    *
470 REM *                      *
480 REM ************************
500 PRINT "NOM DU PROGRAMME ";
510 INPUT X$
520 INPUT "ADR. DEBUT DE ZONE";D
530 L=D:GOSUB 1000
540 POKE 36771,D1:POKE36772,D2
550 INPUT "ADR. FIN DE ZONE  ";D
560 L=D-L:GOSUB 1000
570 POKE36777,D1:POKE36778,D2
580 X$=CHR$(34)+LEFT$(X$,8)+CHR$
(34)+CHR$(0)
590 PRINT "METTEZ LE MAGNETO EN
ENREG. ET  APPUYEZ SUR UNE TOUCH
E."
600 EXEC 58743
610 POKE16918,143:POKE16919,162
620 A=VARPTR(X$)+2
630 T=256*PEEK(A)+PEEK(A+1)
640 A=USR(T)
650 Z$="CLOADM"
660 PRINT "VOTRE FICHIER DEVRA E
TRE CHARGE PAR LA COMMANDE SUIVA
NTE :"
670 Z$=Z$+X$+",ADRESSE DEBUT-"+S
TR$(L)
680 PRINT:PRINT Z$
700 END
1000 D1=INT(D/256):D2=D-D1*256
1010 RETURN
```

EXEMPLE D'UTILISATION DE L'ÉCRAN EN MODE GRAPHIQUE

Ce programme remplit, pour chaque mot de huit bits envoyé au générateur d'écran, celui-ci étant en mode graphique, un écran avec les points correspondant à ces codes. Pour passer à l'écran suivant, on utilise la touche "SHIFT".

1 — le programme Basic qui réserve en haut de mémoire centrale une zone protégée pourle langage machine ;
2 — le programme Basic vous permettant de rentrer dans cette zone protégée le code machine correspondant au programme de génération d'écrans graphiques ;
3 — le programme de générateurs d'écrans graphiques désassemblé.

```
1 REM ************************
2 REM *                      *
3 REM *  RESERVATION DE 55   *
4 REM *  OCTETS POUR LE PRO- *
5 REM *  GRAMME EN LANGAGE   *
6 REM *       MACHINE.       *
7 REM *                      *
8 REM ************************
9 REM
10 POKE16977,200
20 POKE158,200
30 POKE162,200
40 EXEC 63340
50 END
```

```
1 REM ************************
2 REM *                      *
3 REM *  MISE EN PLACE DU    *
4 REM *  PROGRAMME LANGAGE   *
5 REM *  MACHINE A PARTIR    *
6 REM *  DE L'ADRESSE '8FD2  *
7 REM *                      *
8 REM ************************
9 REM
10 I=36818
20 READ X
30 IF X<0 THEN 60
40 POKE I,X
50 I=I+1
60 GOTO 20
80 END
```

```
100 DATA 134,0,206,64,0,167,0,8,
140,68,0,38,248,20,,255,255,9,38
,253,214,3,196,2,38,250,76,77
110 DATA 38,229,134,0,183,191,25
5,222,157,255,66,80,126,247,136,
-1
```

LISTING DU PROGRAMME DESASSEMBLE

```
8FD2 86 00      LDA A,'00
8FD4 CE 40 00   LDX ,'4000
8FD7 A7 00      STA A,('00,X)
8FD9 08         INX
8FDA 8C 44 00   CPX ,'4400
8FDD 26 F8      BNE ,'8FD7
8FDF CE FF FF   LDX ,'FFFF
8FE2 09         DEX
8FE3 26 FD      BNE ,'8FE2
8FE5 D6 03      LDA B,('03)
8FE7 C4 02      AND B,'02
8FE9 26 FA      BNE ,'8FE5
8FEB 4C         INC A
8FEC 4D         TST A
8FED 26 E5      BNE ,'8FD4
8FEF 86 00      LDA A,'00
8FF1 B7 BF FF   STA A,('BFFF)
8FF4 DE 9D      LDX ,('9D)
8FF6 FF 42 50   STX ,('4250)
8FF9 7E F7 6C   JMP ,('F76C)
```

octet à placer sur l'écran (voir tableau des codes du 6874).

remplissage de l'écran

temporisation

teste le bit 1 de l'adresse 03 et boucle tant que la touche "shift" n'est pas enfoncée. "CTRL" passe au code suivant (jusqu'à 'FF)

repositionne l'écran en mode alphanumérique

retour au Basic

JEU "SIMONE"

Ce programme est une variante du célèbre jeu "SIMON". ALICE vous propose des combinaisons successives. Trois niveaux de difficultés sont proposés, qui modifient les vitesses du jeu. Tout d'abord, après la page titre, la règle du jeu apparaît sur l'écran. Puis, après que vous ayez choisi votre niveau de difficulté, le jeu commence.

Structure du programme

lignes 1 à 15	: initialisation du jeu.
lignes 17 à 60	: ALICE vous propose sa combinaison.
lignes 70 à 120	: c'est à vous de jouer.
lignes 130 à 160	: ALICE compare votre proposition à sa combinaison et affiche les différents coups.
ligne 170	: vous avez bien répondu.
ligne 180	: vous avez perdu.
lignes 200 à 240	: fin de la partie.
lignes 850 à 1000	: données correspondant aux dessins d'écran.
lignes 2000 à 3140	: présentation du jeu et choix du niveau de difficulté.
lignes 4000 à 4020	: écran correspondant au gain de la partie.
lignes 4500 à 4630	: écran correspondant à la perte de la partie.
lignes 5000 à 5030	: fin du jeu.
lignes 7000 à 8000	: page titre du jeu.

Ce jeu nécessite l'extension 16 k pour pouvoir fonctionner. Il est toutefois possible de l'adapter à une version de base sans extension en réalisant deux modules :
— un premier programme créera uniquement la page titre, qui sera alors sauvegardée sur la cassette. Cette page titre pourra être sauvegardée grâce à la routine en langage machine donnée par ailleurs ;
— un second programme, dans lequel les lignes 7010 à 7070 et 850 à 1000 seront supprimées, appelera en ligne 7000 la page titre par :

```
7000 CLOADM"ECRAN",16384-512
```

```
1 GOSUB 7000
5 GOSUB 3000
10 DIM L(8),P(8),J(8)
12 FORI=1TO 8:READX:L(I)=X-33:NE
XTI
15 FORI=1TO8:X=L(I)+33:POKEX,48+
1:NEXTI
17 K=0
20 K=K+1
30 FORI=1 TO K:X=RND(8):P(I)=X
50 W=0:GOSUB2000:SOUNDX*16,3*N:W
=1:GOSUB2000
60 NEXTI
70 PRINT@10,"A VOUS...";
80 FOR I=1 TO K
90 N$=INKEY$:IFN$<"1"ORN$>"8"THE
N90
100 X=VAL(N$):J(I)=X
110 W=0:GOSUB2000:SOUND16*J(I),3
*N:W=1:GOSUB2000
120 NEXTI
130 B=0
140 FOR I=1 TO K
150 IF P(I)<>J(I) THENB=1:GOTO16
0
160 NEXTI
162 FOR X=16384 TO 16402:POKE X,
191:NEXTX
165 IF B=0 AND K<8 THEN FOR Z=0
TO 750:NEXTZ:GOTO 20
170 IF B=0THENGOSUB4000:GOTO200
180 GOSUB4500
200 PRINT@448,"UNE AUTRE PARTIE?
 (O/N)"
210 R$=INKEY$:IF R$="" OR(R$<>"O
" AND R$<>"N") THEN 210
220 IF R$="O" THENCLS:GOSUB 3070
:GOTO15
230 GOSUB5000
240 END
850 DATA 16518,16517,16516,16515
,16547,16579,16611,16612,16613,1
6614,16646,16678,16710,16709,167
08,16707
855 DATA 16520,16552,16584,16616
,16648,16680,16712,16456
860 DATA 16714,16682,16650,16618
,16586,16554,16522
861 DATA 16523,139,16555,133,165
56,139,16588,132,16589,136,16557
,135,16558,138,16526,135
862 DATA 16527,16559,16591,16623
,16655,16687,16719
865 DATA 16529,16561,16593,16625
,16657,16689,16721,16722,16723,1
6724,16692,16660,16628,16596,165
64,16532,16531,16530
870 DATA 16726,16694,16662,16630
,16598,16566,16534
871 DATA 16535,139,16567,133,165
68,139,16600,133,16601,139,16633
,133,16634,139
872 DATA 16666,133,16731,16699,1
6667,16635,16603,16571,16539
875 DATA 16548,16542,16541,16573
,16605,16637,16638,16669,16701,1
6733,16734,16735
1000 DATA 16526,16593,16660,1672
1,16782,16715,16648,16587
2000 D=L(K)
2010 IFW=0THENT=143+(X-1)*16:GOT
O 2020
2015 T=128
2020 FORJ=0TO2:POKED+J,T:NEXTJ
2030 POKED+32,T:POKED+34,T FORJ=
64TO66:POKED+J,T:NEXTJ
2040 RETURN
3000 CLS
3010 PRINT TAB(10);"S I M O N E"
:PRINT TAB(10);"----------"
3020 PRINT@96,"ALICE VA VOUS
PROPOSER UNE COMBINAISON DE
8 COULEURS CHOISIES AU HASA
RD."
3025 PRINT"LE JEU CONSISTE A RET
ROUVER CES 8 COULEURS DANS L'ORD
```

```
RE  DANS  LEQUEL ALICE VOUS LES
 A MONTREES"
3030 PRINT@452,"<APPUYEZ SUR UNE
 TOUCHE>";
3040 IFINKEY$="" THEN3040
3050 CLS
3060 PRINT@96,"ENTREZ VOTRE PREN
OM";:INPUTP$
3070 PRINT@160,"NIVEAU DE DIFFIC
ULTE"
3080 PRINT@196,"<1> DIFFICILE"
3090 PRINT@228,"<2> FACILE"
3100 PRINT@260,"<3> TRES FACILE"
3110 PRINT@320,"ENTREZ VOTRE CHO
IX"
3120 PRINT@340,"";:N$=INKEY$:IFN
$<"1"OR N$>"3" THEN 3120
3130 N=VAL(N$)
3135 CLS0
3140 RETURN
4000 PRINT@0,"":PRINT@13,"BRAVO!
...";
4005 W=0
4010 FOR X=1 TO 8:GOSUB2000:FOR
I=1 TO 4:SOUND100,1:SOUND150,1:N
EXTI:NEXTX
4020 RETURN
4500 FOR I=200 TO 100 STEP-2:SOU
NDI,1:NEXTI
4510 PRINT@0,"ALICE A JOUE"
4520 FOR I=1 TO K:X=P(I)
4530 PRINT@13+2*I,X;
4540 W=0:GOSUB2000:SOUNDX*16,N*3
:W=1:GOSUB2000
4550 NEXTI
4555 FOR I=1 TO 1000:NEXT I
4560 PRINT@32,"VOUS AVEZ JOUE"
4570 FOR I=1 TO K:X=J(I)
4580 PRINT@45+2*I,X;
4590 W=0:GOSUB2000:SOUNDX*16,N*3
:W=1:GOSUB2000
4600 NEXTI
4610 PRINT@448,TAB(13);"PERDU !.
.."
4620 FORI=1TO5:SOUND206,2:SOUND2
12,2:NEXTI
4630 RETURN
5000 CLS
5010 PRINT@230,"BYE BYE, ";P$;".
.."
5015 PRINT@448,"";
5020 FOR I=1 TO 1000:NEXT I
5030 RETURN
7000 CLS0
7010 FOR I=1 TO 16:READ X:POKE X
,159:NEXT I
7020 FOR I=1 TO8:READX:POKEX,175
:NEXTI
7030 FOR I=1 TO7:READX:POKEX,191
:NEXTI
7031 FOR I=1 TO 8:READ X,Y:POKE
X,Y+48:NEXTI
7032 FOR I=1 TO 7:READ X:POKE X,
191:NEXTI
7040 FOR I=1 TO18:READX:POKE X,2
07:NEXTI
7050 FOR I=1 TO 7:READ X:POKEX,2
23:NEXTI
7051 FORI=1 TO8:READ X,Y:POKE X,
Y+80:NEXTI
7052 FOR I=1TO 7:READX:POKEX,223
:NEXTI
7060 FOR I=1 TO12:READX:POKEX,23
9:NEXTI
7070 PRINT@450,"(C) A. BONNEAUD
& SORACOM 1984";
7100 FOR I=1 TO 8:X=RND(8):SOUND
 X*16,10:NEXTI
7900 FOR I=1 TO 2000:NEXT I
8000 RETURN
```

JEU DES ANIMAUX

Ce jeu est aussi un excellent divertissement éducatif pour les enfants. Le but du jeu est de penser à un animal et l'ordinateur devra découvrir l'animal auquel vous avez pensé. En début de partie, ALICE ne connaît qu'un seul animal. Mais à chaque fois que vous gagnez, ALICE enregistre vos réponses et acquiert donc de plus en plus de données. De telle sorte que, après plusieurs essais infructueux, elle pourra vous proposer plusieurs noms. Ce programme permet de sauvegarder sur cassette le fichier ainsi constitué afin de le retrouver lors d'une partie ultérieure.

```
1 GOSUB 7000
10 CLEAR 10000
20 DIM T(200,2),T$(200)
22 D1=0:D2=0:K=0
23 A=0:B=0
30 GOSUB 3000
40 CLS:PRINT@140,"VEUX-TU : ";
50 PRINT@195,"<1> COMMENCER UNE
PARTIE";
60 PRINT@259,"<2> REPRENDRE UNE
PARTIE";
70 PRINT@395,"TON CHOIX?";
80 R$=INKEY$:IF R$<>"1" AND R$<>
"2" THEN 80
90 IF  R$="1" THEN GOSUB 1000:GO
TO 110
100 GOSUB 2000
110 CLS:PRINT@ 40,"PENSE A UN AN
IMAL";:I=1
120 PRINT@104,T$(I);"?"
130 R$=INKEY$:IF R$<>"O" AND R$<
>"N" THEN 130
140 IF R$="O" THEN I=T(I,1):GOTO
 160
150 I=T(I,2)
160 IF T(I,1)<>0 THEN GOTO 120
170 PRINT@200,"EST-CE UN(E) ";
180 PRINT T$(I);"?";
190 R$=INKEY$:IF R$<>"O" AND R$<
>"N" THEN 190
200 IF R$="O" THEN PRINT@ 232,"J
'AI GAGNE!!!!":GOTO 330
210 N=N+1:P=P+1
220 PRINT@256,"DE QUOI S'AGIT-IL
";
230 INPUT R$
235 PRINT@288,"DONNE-MOI UNE QUE
STION DISTIN-  GUANT UN(E) ";R$;
```

```
" D'UN(E) ";T$(I)
250 INPUT Q$
255 IF RIGHT$(Q$,1)="?" THEN Q$=
LEFT$(Q$,LEN(Q$)-1)
260 PRINT "QUELLE SERAIT LA REPO
NSE POUR UN(E) ";R$
270 E$=INKEY$:IF E$<>"O" AND E$<
>"N" THEN 270
280 T$(N)=T$(I):T$(N+1)=R$:T$(I)
=Q$
290 IF E$="O" THEN T(I,1)=N+1:T(
I,2)=N:GOTO 310
300 T(I,1)=N:T(I,2)=N+1
310 N=N+1
320 CLS:PRINT@128,"MAINTENANT JE
 CONNAIS ";P;"BETES"
330 PRINT@296,"UNE AUTRE PARTIE?
"
340 R$=INKEY$:IF R$<>"O" AND R$<
>"N" THEN 340
350 IF R$="O" THEN GOTO 110
360 PRINT@320,"VEUX-TU GARDER TE
S REPONSES?"
370 R$=INKEY$:IF R$<>"O" AND R$<
>"N" THEN 370
380 IF R$="O" THEN GOSUB 4000
390 CLS
400 PRINT@266,"BYE BYE ";N$
410 GOTO 10000
1000 N=3:P=2
1010 T$(1)="VIT-IL DANS L'EAU"
1020 T$(2)="CRABE"
1030 T$(3)="CHAT"
1040 T(1,1)=2:T(1,2)=3:T(2,1)=0:
T(2,2)=0:T(3,1)=0:T(3,2)=0
1050 RETURN
2000 CLS
2010 PRINT"PLACE TA CASSETTE EN
LECTURE ET APPUIES SUR UNE TOUCH
E D'ALICE"
2020 EXEC 58743
2030 CLS3:PRINT"LECTURE DU FICHI
ER";
2040 CLOADM"POINTR",149-13
2050 A=256*PEEK(149)+PEEK(150)
2060 B=256*PEEK(153)+PEEK(154)-A
2070 CLOADM"VARIABLE",A-B
2080 A=256*PEEK(155)+PEEK(156)
2090 B=256*PEEK(161)+PEEK(162)-A
2100 CLOADM"CHAINE",A-B
2110 CLS
2120 RETURN
3000 CLS:PRINT@13,"Z O O"
3010 PRINT@45,"-----"
3020 PRINT@96,"LE BUT DE CE JEU
EST DE TROUVER DES NOMS D'ANIMAU
X."
3030 PRINT"TU DOIS PENSER A UN N
OM D'ANIMALET JE VAIS ESSAYER DE
 LE DEVINEREN TE POSANT DES QUES
TIONS AUX- QUELLES TU DEVRAS ";
3040 PRINT "REPONDRE PAR :"
3050 PRINT@296,"'O' POUR OUI"
3060 PRINT@322,"'N' POUR NON"
3070 PRINT@384,"QUEL EST TON NOM
";:INPUT N$
3080 PRINT@455,"PRET A JOUER ";N
$;"?";
3090 R$=INKEY$:IF R$<>"O" THEN 3
090
3100 RETURN
4000 POKE16918,143:POKE16919,162
4005 CLS:PRINT"METS LE MAGNETO E
N ENREG. ET    APPUIES SUR UNE T
OUCHE D'ALICE"
4007 EXEC 58743
4010 POKE 36771,0:POKE36772,149
4020 POKE 36777,0:POKE36778,162
4030 X$=CHR$(34)+"POINTR"+CHR$(3
4)+CHR$(0)
4040 GOSUB 5000
4050 FOR I=1 TO 100-NEXT I
4060 POKE 36771,PEEK(149):POKE 3
6772,PEEK(150)
4070 POKE 36777,PEEK(153):POKE36
778,PEEK(154)
```

```
4080 X$=CHR$(34)+"VARIABLE"+CHR$
(34)+CHR$(0)
4090 GOSUB 5000
4100 FOR I=1 TO 100:NEXT I
4110 POKE 36771,PEEK(155):POKE 3
6772,PEEK(156)
4120 POKE 36777,PEEK(161):POKE 3
6778,PEEK(162)
4130 X$=CHR$(34)+"CHAINE"+CHR$(3
4)+CHR$(0)
4140 GOSUB 5000
4150 RETURN
5000 D1=VARPTR(X$)+2
5010 D2=256*PEEK(D1)+PEEK(D1+1)
5020 CLS 3:PRINT"ENREGISTREMENT
DE ";X$
5030 D1=USR(D2)
5040 RETURN
7000 I=36770
7010 READ X
7020 IF X<0 THEN 7060
7030 POKE I,X
7040 I=I+1
7050 GOTO 7010
7060 RETURN
8000 DATA 206,64,0,255,66,111,20
6,66,0,255,66,113,222,244,255,14
3,242,189,239,79,223,244,198,2,1
89,252,79,254,143,242,223,244
8010 DATA 252,143,242,189,236,22
7,57,-1
10000 END
```

Faire auparavant il faut exécuter le programme de réservation mémoire

Poke 16977,150
Poke 162,150
Poke 158,150
EXEC 63340
puis charger le programme

GESTION DE COMPTE-CHEQUES

Il s'agit là d'un petit programme de gestion de compte-chèque qui vous permet, à chaque utilisation, de sauvegarder sur cassette l'état de votre compte (dates des opérations, libellé, sens de l'opération, montant et solde).
Ce programme met en œuvre bon nombre d'astuces décrites dans le développement de cet ouvrage.

```
10 CLEAR 3000
15 GOSUB 7000
20 DIM D(100),N(100),M(100),S(100),L$(100)
21 REM***INIT. DES VARIABLES****
22 D1=0:D2=0:K=0
23 A=0:B=0
24 REM*****AFFICHAGE DU MENU****
25 CLS
30 PRINT@3,"GESTION DE COMPTE CHEQUES"
35 PRINT@35,"----------"
40 PRINT@78,"MENU"
50 PRINT@ 128,"<1> CREATION DU FICHIER"
60 PRINT@ 160,"<2> SAISIE D'UNE OPERATION"
70 PRINT@ 192,"<3> CHARGEMENT FICHIER CASSETTE"
80 PRINT@ 224,"<4> SAUVEGARDE FICHIER SUR K7"
90 PRINT@ 256,"<5> VISUALISATION OPERATIONS"
100 PRINT@ 288,"<6> FIN"
110 PRINT@ 426,"VOTRE CHOIX?"
120 R$=INKEY$:IF R$="" OR R$<"1" OR R$>"6" THEN 120
130 R=VAL(R$)
140 IF R=6 THEN GOTO 10000
150 ON R GOSUB 1000,2000,3000,4000,5000
160 GOTO 25
990 REM***CREATION DU FICHIER***
995 REM***INIT. DES VARIABLES***
1000 CLS 8
1010 PRINT@264,"CREATION EN COURS";
1020 N0=0
1030 FOR I=1 TO 100
```

```
1040 D(I)=0:N(I)=0:M(I)=0:S(I)=0
:L$(I)=""
1050 NEXT I
1060 RETURN
1990 REM*******SAISIE************
**********D'UNE OPERATION*******
2000 CLS
2020 PRINT@5,"SAISIE D'UNE OPERA
TION";
2030 NO=NO+1
2050 PRINT@ 66,"NUMERO DE CHEQUE
  : ";
2060 INPUT N(NO)
2070 PRINT@130,"DATE OP.(AAMMJJ)
  : ";
2080 INPUT D(NO)
2090 PRINT@194,"LIBELLE OPERATIO
N : ";
2100 INPUT L1$:L$(NO)=LEFT$(L1$,
30)
2110 PRINT@258,"MONTANT OPERATIO
N : ";
2120 INPUT M(NO)
2130 PRINT@330,"<D>EBIT OU <C>RE
DIT";
2140 R$=INKEY$:IF R$<>"D" AND R$
<>"C" THEN 2140
2150 IF R$="D" THEN M(NO)=-M(NO)
:PRINT@ 277,M(NO)
2160 S(NO)=S(NO-1)+M(NO)
2170 PRINT@ 386,"NOUVEL AVOIR
  : ";S(NO)
2180 X$="<C>orrection  <S>uite
<F>in"
2190 PRINT@482,X$;
2200 R$=INKEY$:IF R$="" THEN 220
0
2210 IF R$="C" THEN 2050
2220 IF R$="S" THEN 2000
2230 IF R$<>"F" THEN 2200
2240 RETURN
2990 REM*****LECTURE SUR********
********CASSETTE DU FICHIER*****
*********DES OPERATIONS DEJA*****
**********REALISEES*********
3000 CLS 3
3010 PRINT"REMBOBINEZ LA CASSETT
E PUIS     APPUYEZ SUR UNE TOUCH
E ET METTEZLE MAGNETO EN LECTURE
"
3020 EXEC 58743
3030 CLS3:PRINT@10,"LECTURE DU F
ICHIER";
3034 REM--RESTITUTION POINTEURS-
3035 CLOADM"POINTR",149-13
3038 REM---RESTITUTION TABLE----
-------------DES VARIABLES------
3040 A=256*PEEK(149)+PEEK(150)
3050 B=256*PEEK(153)+PEEK(154)-A
3060 CLOADM"VARIABLE",A-B
3065 REM---RESTITUTION ZONE-----
---------'CHAINES' DE LA PILE----
3070 A=256*PEEK(155)+PEEK(156)
3080 B=256*PEEK(161)+PEEK(162)-A
3090 CLOADM"CHAINE",A-B
3100 RETURN
3990 REM****SAUVEGARDE SUR******
*******CASSETTE DU FICHIER DES**
**********OPERATIONS********
3990 REM---POINT D'ENTREE USR---
4000 POKE 16918,143:POKE16919,16
2
4005 CLS3:PRINT"METTEZ LE MAGNET
O EN ENREG. PUISAPPUYEZ SUR UNE
TOUCHE"
4006 EXEC 58743
4008 REM---SAUVEGARDE DES-------
-------------POINTEURS----------
4010 POKE 36771,0:POKE36772,149
4020 POKE 36777,0:POKE36778,162
4030 X$=CHR$(34)+"POINTR"+CHR$(3
4)+CHR$(0)
4040 GOSUB 6000
4050 FOR K=1 TO 1000:NEXTK
4055 REM--SAUVEGARDE DE LA------
--------TABLE DES VARIABLES-----
```

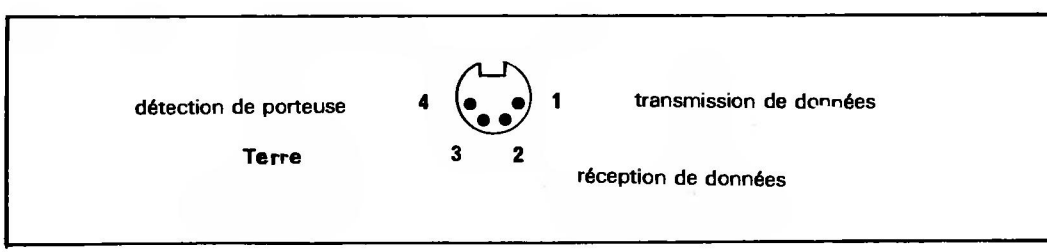
```
4060 POKE 36771,PEEK(149):POKE36
772,PEEK(150)
4070 POKE 36777,PEEK(153):POKE36
778,PEEK(154)
4080 X$=CHR$(34)+"VARIABLE"+CHR$
(34)+CHR$(0)
4090 GOSUB 6000
4100 FOR K=1 TO 1000:NEXTK
4105 REM----SAUVEGARDE DE LA----
------ZONE 'CHAINE' DE LA PILE---
4110 POKE 36771,PEEK(155):POKE 3
6772,PEEK(156)
4120 POKE 36777,PEEK(161):POKE36
778,PEEK(162)
4130 X$=CHR$(34)+"CHAINE"+CHR$(3
4)+CHR$(0)
4140 GOSUB 6000
4150 RETURN
4990 REM***VISUALISATION DE*****
********TOUTES LES OPERATIONS***
**********DEJA ENREGISTREES****
5000 CLS:PRINT@10,"RELEVE DU COM
PTE"
5010 FOR I=1 TO NO
5020 PRINT@ 66,"NUMERO DU CHEQUE
  : ";N(I)
5030 PRINT@130,"DATE OPERATION
  : ";D(I)
5040 PRINT@194,"LIBELLE OPERATIO
N: ";L$(I)
5050 PRINT@258,"MONTANT OPERATIO
N: ";M(I)
5060 PRINT@386,"NOUVEL AVOIR
  : ";S(I)
5070 PRINT@482,"<S> POUR OPERATI
ON SUIVANTE";
5080 R$=INKEY$:IF R$<>"S" THEN 5
080
5090 NEXT I
5100 CLS3:PRINT@263,"PLUS D'OPER
ATIONS";
5110 FOR I=1 TO 2000:NEXTI
5120 RETURN
5990 REM*******APPEL DE LA*******
*******ROUTINE DE SAUVEGARDE****
********EN LANGAGE MACHINE******
6000 D1=VARPTR(X$)+2
6010 D2=256*PEEK(D1)+PEEK(D1+1)
6020 CLS3:PRINT"ENREGISTREMENT D
E ";X$
6030 D1=USR(D2)
6050 RETURN
6990 REM****ON PLACE EN HAUT****
********DE MEMOIRE LA ROUTINE***
**********DE SAUVEGARDE EN****
**********LANGAGE MACHINE***
6992 REM
6993 REM      ATTENTION!!!!!
        CETTE ZONE DOIT ETRE
              PROTEGEE!!!!
6994 REM
7000 I=36770
7010 READ X
7020 IF X<0 THEN 7060
7030 POKEI,X
7040 I=I+1
7050 GOTO 7010
7060 RETURN
7100 DATA 206,64,0,255,66,111,20
6,66,0,255,66,113,222,244,255,14
3,242,189,239,79,223,244,198,2,1
89,252,79,254,143,242,223,244
7110 DATA 252,143,242,189,236,22
7,57,-1
10000 END
```

ANNEXE

SPECIFICATIONS TECHNIQUES

Transformateur	: - entrée	220 V — 50 Hz — 13 VA
	- sortie	10 V — 1,3 A
Composants	: - Microprocesseur	MC 6803 P
	- Générateur contrôleur d'écran	MC 6847
	- RAM 2 circuits 4016 de 2 k chacun	soit 4 k
	- ROM EPROM 2764	8 k
Encombrement	: - dimensions (mm)	51 × 216 × 178
	- poids	850 grammes
Températures	: - Fonctionnement	5 à 40°C
	- Stockage	−20 à 70°C
Taux d'humidité	: - Fonctionnement	40 % à 80 %
	- Stockage	20 % à 90 %

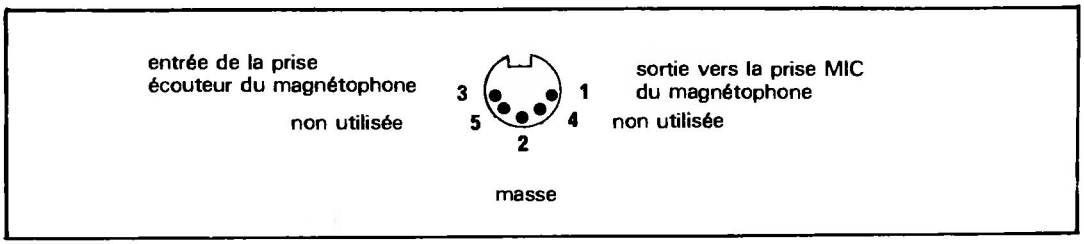
DESCRIPTION DE LA FICHE RS 232 C (entrée/sortie série)



Si vous utilisez une imprimante TANDY, le câble de liaison sera réalisé fil à fil entre les deux connecteurs.
Cette fiche série présente les caractéristiques suivantes :

- vitesse de transfert : 600 bauds ; possibilité jusqu'à 9600 bauds.
- 1 bit de départ ;
- 8 bits de données ;
- 2 bits d'arrêt ;
- pas de parité ;
- impression jusqu'à 132 colonnes ;
- retour chariot automatique en fin de ligne.

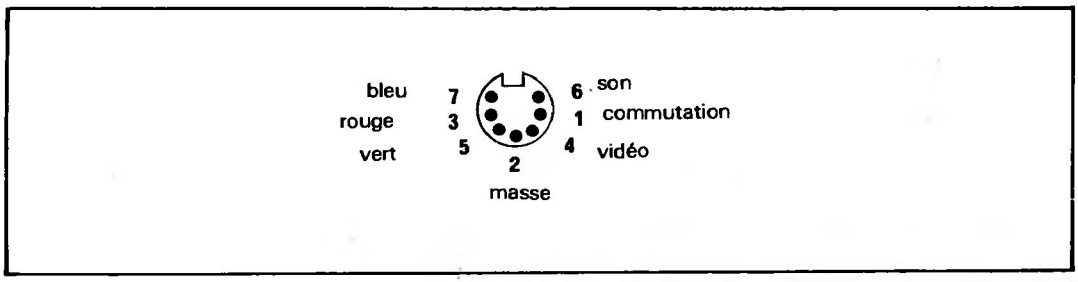
DESCRIPTION DE LA FICHE MAGNETOPHONE



L'interface magnétophone présente les caractéristiques suivantes :

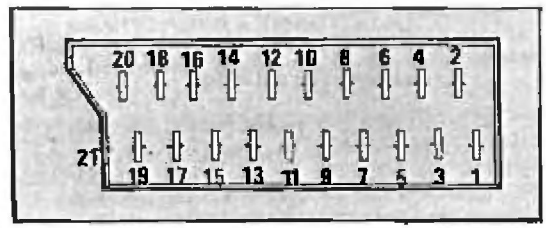
- niveau recommandé en lecture (entrée) : de 1 à 5 V crête à crête à une impédance minimale de 220 Ω ;
- niveau de sortie : 800 mV crête à crête, à 1 000 Ω.

DESCRIPTION DE LA FICHE DE SORTIE TV



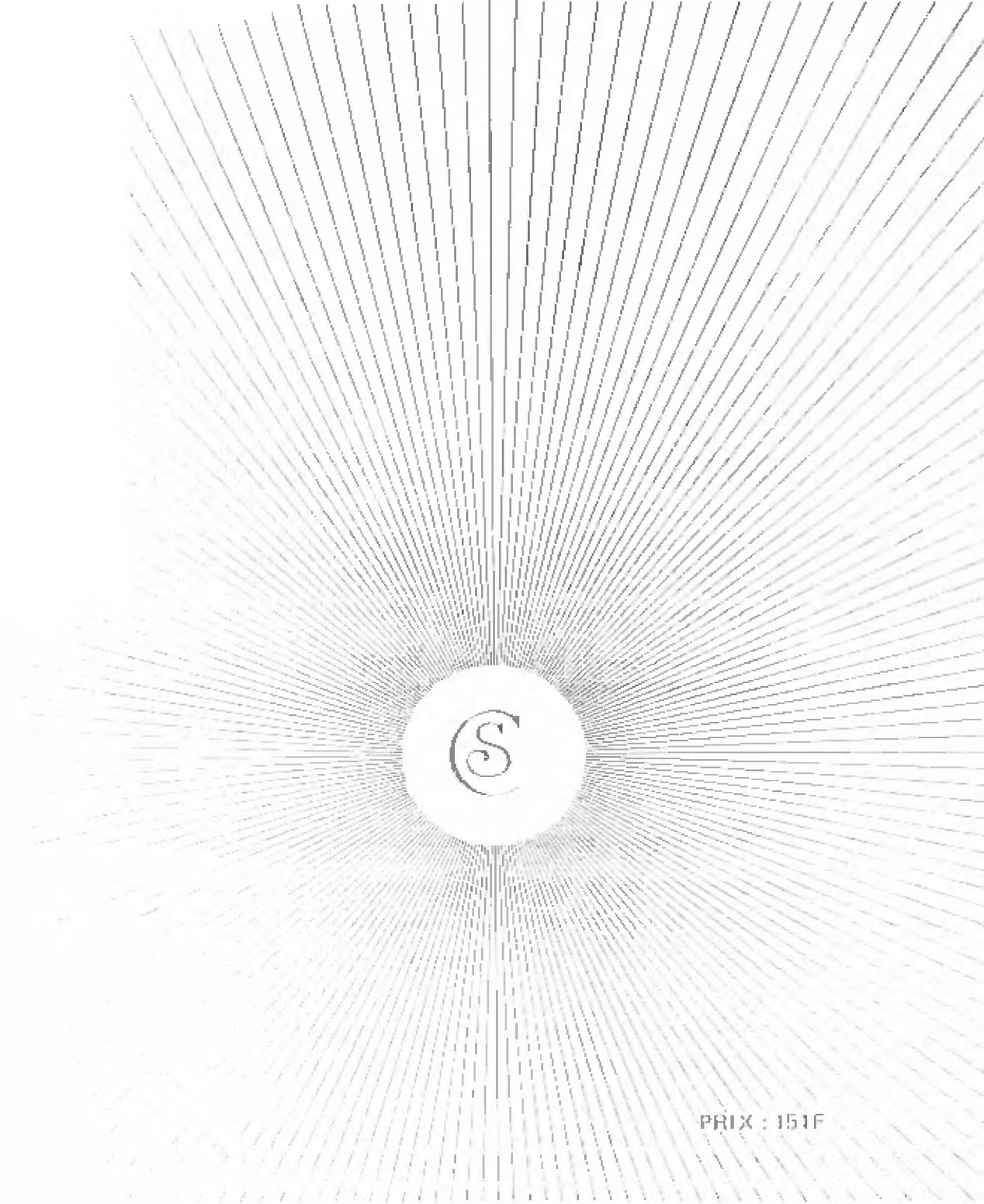
DESCRIPTION DE LA PRISE PERITEL

6 = son
7 = bleu
8 = commutation lente
11 = vert
15 = rouge
16 = commutation rapide
17 = masse
18 = vidéo



SOMMAIRE

Composition : FIDELTEX
Maquette : SORACOM
Impression : VAN DEN BRUGGE
Dessin de couverture : T . BERTRAND
N°d ' éditeur : 039
Dépôt légal :4 éme trimestre 1984



PRIX : 151F