A.D.E.S.

Manuel d'utilisation

A.D.E.S. a été écrit pour donner aux passionnés de micro-informatique sur AMSTRAD, les pleins pouvoirs sur leur machine. En effet, ce logiciel ne permet pas seulement de programmer en assembleur, il est aussi un outil d'exploration, de debuggage et de bidouillage.

Les principales caractéristiques de ce logiciel est qu'il ne vérifie rien ; A.D.E.S. ne refusera jamais d'assembler dans les vecteurs systémes, dans la mémoire écran... Et le moniteur permettra de travailler à n'importe quelle adresse en RAM. Par conséquent, il offrira le maximum de possibilités aux programmeurs.

Ce manuel présentera successivement l'éditeur de texte, l'assembleur et le moniteur. Ce n'est pas un outil d'initiation à l'assembleur ; il faut simplement savoir que, comme en BASIC, les nombres hexadécimaux sont précédés de "&".

L'EDITEUR DE TEXTE.

Il permet la construction d'un programme source que l'on assemblera avec l'assembleur et que l'on éxécutera avec le moniteur.

La syntaxe des lignes est la suivante :

N° ligne Label. Mnémonique Opérandes ; commentaires

Exemple :

```
100 Affiche. CALL &BB5A ; Aff car. dans A
```

Le texte source débutera à l'adresse &2500 et se terminera à une adresse qui dépendra de la longueur du texte entrée. Il faudra veiller à faire de fréquentes sauvegardes de texte , car l'utilisation du désassembleur pourra le détruire si certaines régles ne sont pas suivies.
Sous Editeur de texte, par la commande I , on obtient la derniére adresse du texte.
On veillera donc sous désassembleur à :

- ne pas poker dans le texte
- ne pas faire de copie de zone de mémoire dans le texte
- ne pas utiliser la commande F

(remplir zone) dans le texte
- ne pas charger de fichier dans le texte
- ne pas fabriquer de source sur un texte

On peut noter qu'en l'absence de texte, ces régles sont inutiles.

voici la liste des commandes possibles sous éditeur :

FONCTION :	COMMANDES :
. lister le texte	L,N° de ligne L,label
. demander le n° de la derniére ligne	U
. réinitialiser le texte (commande NEW)	N
. quitter l'éditeur	Q
. catalogue de la disquette	C
. information sur le texte	I
. visualisation des labels du texte	V

. détruire un bloc de lignes	D,N° ligne début, N° ligne de fin
. renuméroter les lignes du texte	R,N° ligne début, incrément
	R,incrément
. obtenir numérotage automatique	A,N° de ligne, incrément
. des lignes à entrer	A,N° de ligne
. charger un programme source	G,nom programme
. sauver le texte édité	S,nom programme

Lorsque l'on branche l'éditeur, la mémoire n'est pas réinitialisée ; s'il y a un texte en mémoire, vous pouvez le lister, sinon les lignes devront être entrée au clavier suivant la syntaxe indiquée.

LISTER LE TEXTE :

Cette fonction permettra de donner un listing à l'écran ou sur l'imprimante du texte en mémoire.

L,N° de ligne ou L,label

Un message d'erreur s'affichera si le numero de ligne ou le label n'existe pas

Interruption : <esc> arrêt sur un page
<esc><esc> fin

NB : le dernier numéro lister est garder en mémoire. Seule la commande L permet alors d'obtenir un nouvelle liste.

NUMERO DE LA DERNIERE LIGNE :

Le numero de la dernière ligne s'obtient par la commande U

REMISE A ZERO DU BUFFER DE TEXTE :

La commande N (NEW) élimine le texte de la mémoire.

QUITTER L'EDITEUR :

On retourne au lanceur par la commande Q.
Le fait de quitter l'éditeur ne modifie en rien le texte.
Lorsque l'on revient sous l'éditeur, le texte est tel que lorsque vous l'avez quitté.

CATALOGUE DE LA DISQUETTE :

La commande C donne le catalogue de la disquette dans le A ou B.
NB : pour changer de drive, revenir sous BASIC et taper B:

INFORMATION SUR LE TEXTE :

Cette fonction I donne la longueur et la dernière adresse du texte ; la quantité de mémoire libre est aussi indiquée.

VISUALISATION DES LABELS DU TEXTE :

La commande V donnera la liste des labels du texte avec leur numéro de ligne. Si l'imprimante est aussi connectée, la sortie se fera également sur l'imprimante. On suspend le défilement des labels avec <esc> et on l'arrête par deux appuis consécutifs sur <ESC>.

DESTRUCTION D'UN BLOC DE LIGNES :

La commande D,10,130 détruit toutes les lignes de la N° 10 à la N° 130 incluse.
ATTENTION : un message d'erreur apparaît à l'ecran si les lignes n'existe pas

RENUMEROTATION DES LIGNES DU TEXTE :

R,10,30 renumérote les lignes du texte. La première ligne aura le n°10, la seconde le N° 40...

Si vous n'indiquez pas l'incrément il sera par defaut égal à 10.
Si vos numéros de ligne dépassent 65535, la commande R,1,1 s'effectuera automatiquement.

ENTREE AUTOMATIQUE DES LIGNES DU TEXTE :

Cette commande affiche les numéros de lignes directement.

A,N° de ligne,Increment

Si le pas d'incrément est omis, il sera par défaut égal à 10.
Si les lignes ne sont pas éffacées en mode AUTO, il peut arriver qu'une ligne déjà existante soit affichée, alors les lignes seront décalées de 1 vers le bas.

SAUVER LE TEXTE :

S,Nom sauvera sur disquette le fichier texte

CHARGER UN TEXTE :

G,NOM

Cette fonction est équivalente au MERGE du basic.
Si vous charger un texte alors qu'il y en déjà un, le nouveau texte sera placé à la suite du premier et les numeros de lignes seront réinitialisés.
Si aucun texte n'est présent en mémoire il sera alors simplement chargé.

LES MESSAGES D'ERREURS DE L'EDITEUR :

- **Erreur de syntaxe** : s'affiche lorsque la commande est inexistante ou lorsque les paramétres de l'instruction sont incorrectes

- **Ligne inconnue** : s'affiche si vous demandez une ligne inexistante.

- **N° de ligne incorrecte** : s'affiche si vous dépassez le nombre de lignes autorisées Une renumérotation des lignes est alors lancée.

. Buffer plein : le texte atteint la longueur maximale.

. Label inconnu : s'affiche quand un label recherché n'existe pas.

. Mauvais fichier : s'affiche si vous charger un texte qui n'a pas été créé par. cet éditeur

. Fichier trop long : s'affiche si la jonction de deux fichier dans le buffer dépasse la capacité de celui-ci.

. label incorrect : la taille maximum d'un label est de huit caractéres.

. Mnémonique Incorrect : le mnémonique de la ligne n' a pas été reconnu.

L'ASSEMBLEUR

Ce programme transformera le programme source crée avec l'EDITEUR en un code objet executable par l'AMSTRAD. Les directives de l'assembleur sont les suivantes :

- ORG
- LOAD
- ENT
- DB
- DW
- DS
- EQU

DIRECTIVES ORG :

Placé dans un programme ORG fixe l'adresse où est assemblé le programme source. L'adresse transmise avec ORG devra correspondre à une zone libre de la RAM. Les adresses renvoyant à une zone où est situé GT-Assembleur ou le programme source ne sera pas accepté.
L'assembleur peut assembler n'importe où dans la RAM ou dans la mémoire écran : la

source peut prendre toute la zone de &2500 à &A000. Le programmeur assemblera le texte dans la RAM écran en &C000.
Il faut par conséquent prendre toutes les précautions utiles (sauvegarde du texte...) avant d'assembler dans les vecteurs de l'AMSDOS car il y a un risque de plantage systéme.

```
exemple :
 10   org &4000  ; origine en &4000
 20   LD A,2
 30   JP 1BC0E   ; mode 2 d'écran
```

Avec le désassembleur, on reliera en &4000 :

```
    &4000  3E,02     LD A,2
    &4002  C3,0E,BC  JP BC0E
```

LA DIRECTIVE LOAD :

Cette instruction devra être placé juste après ORG. Placée avant, elle sera annulée est placée plus loin dans le programme et celui-ci n'aurait alors plus aucun sens.

LOAD fixe l'adresse où le code objet sera chargé et executé.

Par exemple, si on assemble une routine dans la mémoire écran et que l'on veuille que celle-ci soit executable en &4000 on écrira au début du programme :

```
10  ORG &C000
20  LOAD &4000
```

LA DIRECTIVE ENT :

Cette commande est utile pour la sauvegarde des programmes. Placée dans le code source, ENT indiquera à l'assembleur le point d'entrée du programme.

LA DIRECTIVE EQU :

EQU initialise un label avec un entier compris entre 0 et 65535 (0 et &FFFF).

```
Exemple :  10 label. EQU &1234 ; label=
                                  &1234
```

A chaque fois que l'assembleur rencontrera la chaine de caractéres "label", il prendra en compte l'entier &1234.

```
20  LD hl,label  ;hl=&1234
```

LA DIRECTIVE DS :

DS signifie définition Space. Il s'agira donc de définir un espace mémoire d'une certaine longueur qui servira à des applications spécifiques.

exemple : 10 DS 100

Définira un espace libre de 100 octets

LES DIRECTIVES DB et DW :

DB et DW signifient respectivement Définition Byte et Définition Word. Leurs opérandes sont multiples :

DB "chaine de caractéres"
OU
DW "chaine de caractéres"
pokera à la suite les caractéres ASCII correspondants.

DB octet (entier de 0 à 255) pokera l'octet correspondant

DB label pokera le poids faible à l'adresse d'assemblage.
(label-16*int(label/16))

DB "n" +DD pokera la valeur ASCII du caractére n + un octet dd. Si la somme dépasse 255, le poids faible sera poké.

On peut aligner tous ces opérandes en les séparant par des virgules.
Exemple : DB &12,100,"A+10",label

Toute cette instruction reste valable pour l'instruction DW, mais l'assembleur pokera le poids fort puis le poids faible de l'entier entrée.

LES MESSAGES D'ERREURS DE L'ASSEMBLEUR :

- **erreur de syntaxe** : la ligne à assembler comporte une erreur dans les opérandes ou sa syntaxe n'est pas permise en assembleur Z80.

- **déplacement incorrect** : dans les instructions de saut relatif (JR, DJNZ, JRcon), certaines limites ne doivent pas être dépassées.

- **mauvaise origine** : l'assembleur ne peut pas assembler sur lui-même ou sur le texte. Dans ce cas utiliser la commande LOAD.

. label indéfini : la chaîne de caractères en opérande dans une instruction n'a pas été définie auparavant comme étant un label.

. labels similaires aux ligne _ et ¯ : l'assembleur vérifiera que vous n'avez pas entré deux fois le même label.

LE MONITEUR

Il a les possibilités suivantes :

FONCTION	
. Désassembleur	D,adresse
. Exécuter pas à pas	T,adresse
. Interprétation de code objet	I,adresse
. Vue globale de la mémoire	M,adresse
. Copie de zones de mémoire	C,adr1,adr2, longueur
. Remplissage d'une zone mémoire	F,adr,long, valeur
. tranformer du code en source	N,adr1,adr2, adr buffer
. brancher une ROM	O,(A=AMSDOS) (D=dos) (B=basic) (E=externe)
. Poker	P,adr,octet
. Voir/réinitialiser ROM	A
. Sauver du code objet	S,adr1,adr2, entrée
. Charger un programme	G,nom,adr

. reloger le moniteur	L,adresse
. Executer un programme avec point d'arrêt	R,adr,adr point arret

DESASSEMBLEUR :

Pour désassembler du code à partir d'une certaine adresse, il faudra effectuer la commande suivante :

D,adresse (entier entre 0 et 65535)

Entre 0 et &3FFF, on pourra désassembler soit la RAM, soit la ROM.

Entre &C000 et &FFF, on pourra désassembler soit la mémoire écran, soit le basic, soit le DOS. Pour cela, il faudra brancher les ROMS ou réinitialiser. Si l'imprimante est connectée et active, le code sortira sur l'écran et sur l'imprimante.

On quitte le désassembleur par <ESC>.

EXECUTION PAS A PAS :

cette commande (T,adr) provoque l'éxecution pas à pas du code objet à partir de l'adresse spécifiée. On pourra voir :

- l'instruction désassembler
- l'état de tous les registres, flags et la pile.
- l'état des RAMS et des ROMS.

On quitte le mode pas à pas par <ESC>

INTERPRETATION DU CODE OBJET :

I,adr provoque l'execution du code objet à l'adresse en entrée.
Cette fonction est ralentie par l'interprétati et peut-être arrétée par la touche <ESC>.
Il faut noter que l'interpréteur ne vérifie rien. Ceci pourra mener à des plantages du systéme si par exemple il execute son auto-destruction . En outre, l'interpreteur utilisant certaines routines de l'AMSDOS, il peut arriver qu'il entre en conflit avec lui-même s'il interprete ces mêmes routines : le programme s'arrête alors.
Enfin, l'interpretation des entrées-sorties

est évidemment impossible. Certains messages d'erreurs et il vaut mieux sortir par <ESC>.

VUE GLOBALE DE LA MEMOIRE :

Cette commande (M,adr) donne un aperçu complet de la mémoire avec la valeur héxadécimale des octets et la valeur des caractéres ASCII associés.

COPIE DE ZONES DE MEMOIRE :

Le déplacement de 100 octets à l'adresse &4000 vers l'adresse &5000 est parfois utile. La commande C,adr1,adr2,longueur réalisera cette fonction. Il faut préciser que le moniteur ne vérifie rien, on pourra planter le systéme. La copie de zones de mémoires pourra se faire partout sauf sur le moniteur.

REMPLISSAGE D'UNE ZONE DE MEMOIRE :

La commande F,adr,longueur,octet remplira la zone entre l'adresse et adresse + longueur par l'octet fourni en entrée. Toutes les adresses sont possibles, sauf celles du moniteur.

TRANSFORMER DU CODE EN SOURCE :

Il peut être utile de mettre une zone de mémoire sous forme de texte source. Cette commande mettra sous le texte source la commande du désassembleur :

N,adr1,adr2,Buffer

Adresse1 et Adresse2 representent l'adresse de départ et l'adresse d'arrivée du code à tranformer. Le moniteur demande ensuite le nom du programme : il le transformera et le sauvera. Pour pouvoir relire immédiatement le source sous éditeur de texte, il faudra prendre comme adresse buffer &2500.

Si le buffer est trop petit, ou situé à des adresses incompatibles, un message d'erreur sera lancé.

BRANCHER UNE ROM :

Le désassembleur et le mode MAP donne des octets correspondants à la RAM. On peut visualiser ceux de la ROM en branchant celle-ci :

0,A branche L'AMSDOS

Un désassemblage à l'adresse 0 donne le décodage de la ROM

0,B branche le BASIC

Un désassemblage à l'adresse &C000 donne le listing du BASIC

0,D branche le DOS

VISUALISER ET REINITIALISER LES ROMS :

La commande A montre la configuration actuelle. L'appui sur <esc> annulera la commande de réinitialisation. L'appui sur n'importe qu'elle touche provoque la réinitialisation, c'est à dire le branchement des RAMS.

SAUVER DU CODE OBJET :

La syntaxe de cette instruction est :

S,nom,adresse de début, adresse de fin, point d'entrée

Cette commande effectue le SAVE du Basic pour une zone de mémoire.

CHARGER UN PROGRAMME :

Comme en basic, on peut charger un fichier binaire en mémoire. L'instruction "G,nom,Adresse" chargera à l'adresse indiquée le fichier binaire de nom "NOM". Le moniteur verifiera s'il ne se détruit pas en chargeant le programme, les mauvaises adresses seront rejetées.

RELOGER LE MONITEUR :

Le moniteur est entiérement relogeable. Il se trouve initialement à l'adresse &420. Si le programmeur désire désassembler, tester et observer une routine qui tourne dans la zone d'adresses du moniteur, il peut déplacer le programme moniteur dans la mémoire.

La commande L,&4000 logera le moniteur à l'adresse &4000. On pourra effectuer toutes les commandes du moniteur et, par exemple, charger et examiner une routine en &420.

Il faut par ailleurs préciser que le moniteur étant complétement indépendant du reste du logiciel, celui-ci ne vérifiera

pas la présence de texte ou de quoi que ce soit d'autre à l'endroit où il va se reloger. Par conséquent, une grande prudence est nécessaire avant d'effectuer ce genre de commande.

EXECUTER UN PROGRAMME AVEC POINT D'ARRET :

R,&4000, effectue un CALL classique comme on peut le faire en Basic.
R,&4000,&4005 appelera l'adresse &4000 et lorsque PC (point counter) sera en &4005, il y aura un retour dans le mode pas à pas du moniteur.

R,adresse de départ,adresse de Break

POKER :

Effectue la commande POKE du Basic.

P,adresse, octet

Le moniteur ne pokera pas sur lui-même.

LES MESSAGES D'ERREURS DU MONITEUR :

. **Commande inconnue** : apparait lorsque le désassembleur ne reconnait pas la commande entrée par l'utilisateur.

. **Erreur de syntaxe** : elle concerne essentiellement les données en entrée des commandes. Les causes peuvent être multiples :

- erreur de frappe ex : &9G3A
- Nombre d'opérandes incorrect ex : d,100,1

. **Adresse incorrect** : le moniteur ne peut pas se détruire en pokant sur lui-même.

ANNEXE 1 :

Caractéristiques des fichiers de GT-Assembleur :

ADES6128.BAS : Lanceur basic pour cpc 6128

ADES664.BAS : Lanceur basic pour cpc 664

ADES6128.BIN : Noyau pour cpc 6128

ADES664.BIN : Noyau pour cpc 664

GTEDIT.BIN : Editeur de texte

. Adresse de départ	&420
. Longueur	&1510
. Point d'entrée	&420

GTASSM.BIN : Assembleur

. Adresse de départ	&420
. Longueur	&1600
. Point d'entrée	&420

GTDESA.BIN : Moniteur

- Adresse de départ &420
- Longueur &2090
- Point d'entrée &420

NB: Si on charge le moniteur en &7000 et que l'on sauve le programme ainsi relogé (S,nom,&7000,&2090) on obtient un programme dont le point d'entrée est en &7000

ANNEXE 2 :

Le vecteur de l'éditeur de ligne est le suivant :

&BD3A	pour CPC 464
&BD5B	pour CPC 664
&BD5E	pour CPC 6128

Le logiciel comporte deux lanceurs :

ADES6128.BIN et ADES664.BAS.
Sur CPC 6128 le programme pourra être lancé par l'un ou l'autre.

Par contre, sur un CPC 664 ou 464, le logiciel ne pourra être lancé que par ADES664.BAS
Il faudra modifier les trois modules suivants :

GTEDIT.BIN
GTASSM.BIN
GTDESA.BIN

afin d'adapter le vecteur de l'éditeur de ligne.

On fera alors le programme suivant :

```
10 memory &3fff
20 input "cpc 464 (1) ou 664 (2)";cpc
30 if cpc=1 then addfai=&3E
            else addfai=&5B
40 load "gtedit.bin",&4000
50 poke &4027,addfai
60 save "gtedit",b,&4000,&1510
70 load "gtassm.bin",&4000
80 poke &4000,addfai
90 save "gtassm.bin",b,&4000,&1600
100 load "gtdesa.bin",&4000
110 poke &5df3.addfai
120 save "gtdesa.bin",b,&4000,&2090
```

PRESENTATION DE DEBUGG

ATTENTION CE LOGICIEL NE TOURNE QUE SUR 6128

Cet utilitaire s'adresse aux personnes utilisant le langage machine Z80, aussi bien aux débutants qu'aux professionnels. Il peut à la fois faciliter la mise au point d'un programme en langage machine et aider à la comprehension d'un programme inconnu.

Il posséde un trentaine de commandes qui en font un utilitaire relativement complet. Il permet entre autre, une utilisation trés approfondie du lecteur de disquettes.

DEBUGG permet de pouvoir suivre le fonctionnement de n'importe quel programme en langage machine grâce à certaines de ces particularités :

- Il n'occupe aucune place en mémoire, étant entiérement résident dans les 64 K supplémentaire du 6128. Ceci permet donc d'utiliser les 64 K courants comme on l'entend, sans probléme de superposition.

- Il est autonome. DEBUGG n'utilise les routines systémes que pour la gestion des disquettes. On peut donc utiliser les parties de la RAM réservées au système sans altérer son fonctionnement. Ce n'est que lors d'un accès aux disques qu'il est nécessaire que la RAM système soit intacte.

- Il peut executer pas à pas toutes les instructions du Z80. Il connait même celles que l'on ne trouve dans aucun livre, qui utilisent par exemple les semi-registres d'index IX et IY (XH,XL,YH ET YL). Ou bien encore, les codes CB 30 à CB 37 qui correspondent à un instruction de décalage inconnue que j'ai baptisée SLL r: C <- registre r <-1

- Il concerve en mémoire l'écran de travail. Cet écran est réaffiché chaque fois qu'une commande est susceptible de le modifier ou alors, ces modifications sont redirigées vers la mémoire tampon. De même, toutes les modifications de couleur ou de structure.

VIGNOUD Stéphane.

LES COMMANDES DE DEBUGG

Certaines commandes doivent être suivies de parametres. Si ceux-ci sont incorrects, DEBUGG les redemande un par un, en spécifiant à chaque fois la nature du parametre demandé.

Les valeurs qui sont éventuellement proposées dans la fenêtre de droite sont celles prises par défaut.

Une valeur qui doit être rentrée sous forme héxadécimale, doit être suivie de la lettre H (ex : BB06H).

Un paramétre peut être le résultat d'une opération ; dans ce cas, les priorités de signe ne sont pas respectées.

Dans la liste qui suit, les paramétres entre parathéses sont facultatifs.

RETURN	Visualise l'ecran de travail
--> <--	Permet de modifier le contenu des registres ou de choisir une option, validée ensuite par return
ESC	Arrête la commande en cours et attend une nouvelle commande
nb	Affiche nb sous la forme hexadécimale, décimale et binaire
A(add),long,oct	Applique un ET logique entre chaque octet de la zone mémoire commencant par ADD, de longueur LONG et l'octet OCT. L'adresse de début par défaut est la première affichée dans la fenêtre.
B(add)	Positionne l'adresse d'un tampon de 22 octets en RAM utilisé lors des commandes E et J. Ce tampon ne doit pas se trouver entre les adresses 3FEAH et 8000H.
C	Affiche le catalogue de la

disquette du lecteur courant.

D(add) Désassemble la mémoire à partir de ADD, ou de la première adresse de la fenêtre :

Deplace le pointeur

ESPACE Place un BREAK à l'adresse pointée. Lorsqu'on lance une execution, celle-ci est stoppée par un break. Pour enlever le break il faut de nouveau appuyer sur ESPACE.

RETURN Ramène l'adresse pointée en haut de la fenêtre

Q Sort du mode desassemblage

E(add),(break1,break2,...) exécute un programme à partir de add ou par défaut, à l'adresse contenue dans le registre PC. Les adresses

sont des adresses à parit desquelles l'exécution est stoppée.

P(add),(oct1,oct2,...) recherche la chaîne d'octet en mémoire, à partir de add. Si la chaîne d'octet n'est pas spécifiée, c'est la précédente qui est prise en compte. L'adresse de début de la recherche est par défaut la première affichée dans la fenêtre.

I(deb),fin Sortie sur l'imprimante de la zone mémoire allant de déb à fin. Si l'option DESA est sélectionnée, la sortie se fera sous la forme d'un désassemblage, sinon ce sera les codes et les caractères ASCI

J Execute une instruction de la forme CALL,CALL2,... Cette instruction doit être pointée par le registre PC.

K Efface tous les BREAK

L Lit un fichier sur le lecteur courant

M(add) Mode modification, à partir de add ou de la première adresse de la fenêtre. Ensuite :
RETURN :permet de taper directement des car. ASCII. Retour à la normale en tapant de nouveau RETURN
Q : quitte le mode modification

N(add),long Applique un Non logique (NOT) à chaque octet de la zone mémoire indiquée. L'adresse de début par défaut, est la première adresse de la fenêtre.

O(add),long,octet applique un OU logique

P(add),(nb) Exécute pas à pas le nb d'instruction du Z80, à partir de add ou de l'adresse contenu dans PC. L'exécution s'arrête sur un break. Une instruction qui planterait DEBUGG, (commutation de blocs de RAM) n'est pas exécutée et le message "INSTRUCTION NON EXECUTEE" s'affiche. Toutes les modifications de l'écran sont redirigées ou mémorisées.

Q Quitte DEBUGG. Le retour s'effectue en tapant DEBUGG

S Sauve un fichier de type binaire (2)

Tsource,long,dest Déplace la partie de la mémoire commençant à la source et de longueur long à l'adresse dest

U(d) Positionne d (A ou B) comme lecteur courant. Si d n'est pas spécifié, le lecteur courant actuel est afiché.

V Visualise et permet de modifier les registres du CRT et du GATE ARRAY. Les déplacements se font grâce aux touches du curseur.

X(add),long,oct applique un OU exclusif XOR

Z Efface un fichier sur le lecteur courant.

DI DEBUGG interdira les

interruptions lorsqu'il lancera une exécution (commandes J ou E) ou lorsqu'il affichera l'écran.

EI — DEBUGG autorisera les interruptions

LS — Lit un ou plusieurs secteurs :
Adresse = adresse de chargement du secteur
Secteur = numéro du 1er secteur
Piste = numéro de la première piste
Lecteur = numéro du lecteur A=0 et B=1
Nombre = nombre de secteurs à lire
Si nombre > 1 :
1er sect=numéro du 1er secteur de chaque piste
S/piste=nombre de sect. par piste

SS — Sauvegarde de secteur;

CP — Cherche une piste. Place la tête de lecture sur la piste voulue.

FP — Formate une ou plusieurs

pistes. Lorsque tous les secteurs sont indiqués, entrez un numéro de secteur supérieur à 255.

NOTA BENE : Dans un paramètre, il est possible d'indiquer le contenu d'un regitre, en mettant un R puis le nom du registre.
exemple : RHL, RIX ou RPC