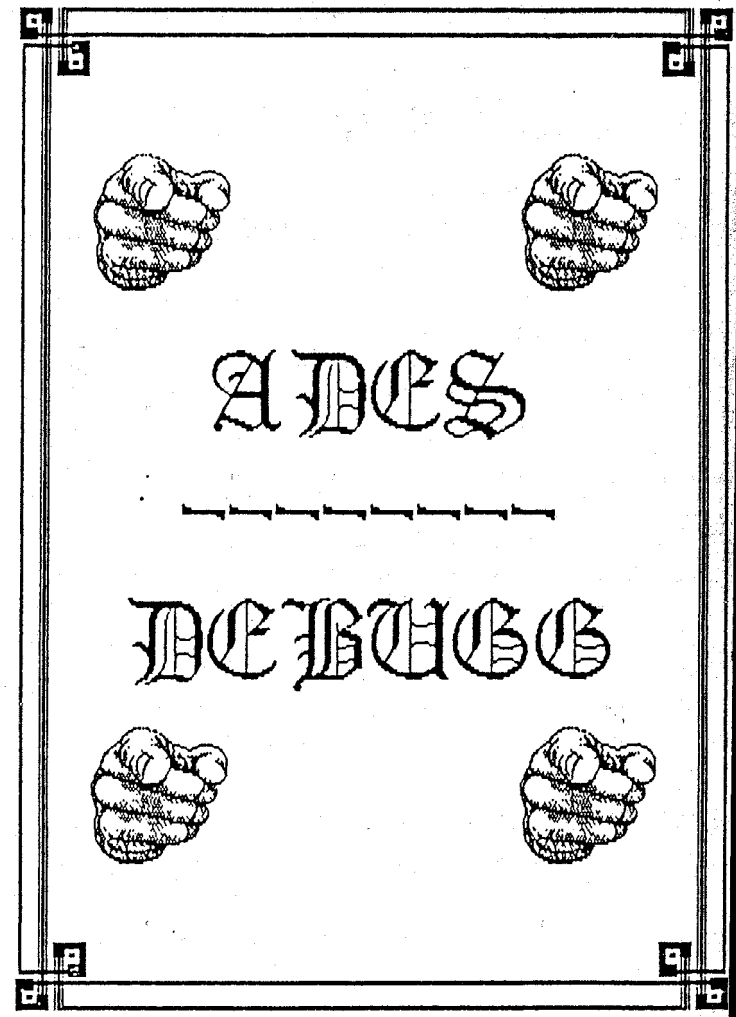
ADES
DEBUGG

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 débuggage et de bidouillage.

Les principales des caractéristiques de ce logiciel est qu'il ne vérifie rien ; A.D.E.S. ne refusera jamais d'assembler dans les vecteurs systèmes , 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 "&".

AVERTISSEMENT:

Ce logiciel se lance par RUN "ADES6128 ou par RUN "ADES664 sur un CPC 6128. Pour un CPC 664 ou 464 , il faudra se reporter à l'annexe 2 et le programme pourra se lancer par RUN "ADES664.

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 :

No de ligne Label. Mnémonique opérandes ;commentaires

Exemple: 100 affiche. CALL &BB5A ;Affiche le caractère 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 du texte , car l'utilisation du désassembleur pourra le détruire si certaines règles ne sont pas suivies.

Sous éditeur de texte , par la commande I , on obtient la dernière adresse du texte. On veillera donc sous désassembleur a

- ne pas poker dans le texte.
- ne pas faire de copie de zones de mémoire dans le texte.
- ne pas utiliser la commande F (remplir zone) dans le texte.
- ne pas reloger le moniteur 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 règles sont inutiles.

Voici la liste des commandes possibles sous éditeur :

FONCTION	COMMANDE
Lister le texte	L,numéro de ligne L,label
Demander le numéro de la dernière 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,No de la première ligne,No de la dernière ligne
Renuméroter les lignes du texte	R,No de la première ligne, incrément R,incrément.
Obtenir automatiquement les numéros des lignes à entrer	A,No de ligne ,incrément
Charger un programme source	G,Nom du programme
Sauver le texte édité	S,nom du programme

Lorsqu'on branche l'éditeur,la mémoire n'est pas réinitialisée ; si il y a un texte en mémoire , on peut le lister sinon on entre des lignes au clavier avec la syntaxe indiquée ci-avant.

LISTER LE TEXTE.

Cette fonction permettra de donner un listing à l'écran ou sur l'imprimante (si celle-ci est branchée) du texte présent en mémoire. La syntaxe de cette instruction est la suivante :

L,Numéro de ligne ou L,label

Si le numéro ou le label n'éxiste pas , un message d'erreur sera lancé.

On interrompt le listage par <ESC> et on l'arrete définitivement par deux <ESC> consécutifs.

NB: Le dernier numéro listé est retenu. La commande L seule permettra donc de relister le texte comme précédemment.

NUMERO DE LA DERNIERE LIGNE.

Le numéro de la dernière ligne s'obtient par la commande U.

REMISE A ZERO DU BUFFER DE TEXTE.

Pour éliminer un texte de la mémoire , on éffectue la commande N (New).

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 éditeur , le texte est tel que lorsqu'on l'a quitté (si on ne l'a pas modifié extérieurement).

CATALOGUE DE LA DISQUETTE.

La commande C donne le catalogue de la disquette dans le drive A ou B .
NB: Pour changer de drive , il faut revenir sous BASIC et entrer la commande 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 connectée , la sortie se fera également sur l'imprimante.
On suspend le défilement des labels avec <ESC> et on l'arrete par deux appuis consécutifs sur <ESC>.

DESTRUCTION D'UN BLOC DE LIGNES.

L'entrée de 'D,10,130' détruira toutes les lignes comprises entre 10 et 130 incluses. Attention , cette commande est irréversible. Les numéros des lignes mentionnées devront exister, sinon un message d'erreur sera lancé.

RENUMEROTATION DES LIGNES DU TEXTE.

R,10,30 renumérotera les lignes du texte. La première ligne aura le numero 10 , la deuxième 40 ...

On suivra la règle : R,numéro de la première ligne,incrément. Si on ne précise pas l'incrément , le logiciel prendra 10. Il peut arriver si le texte est très long qu'un numéro de ligne dépasse 65535. Dans ce cas , le logiciel le précisera et effectuera la commande R,1,1.

ENTREE AUTOMATIQUE DE LIGNES DE TEXTE.

Cette commande affiche a la place du programmeur les numéros des lignes à entrer. La syntaxe de cette fonction est :
A,numéro de ligne , incrément.
Si on omet l'incrément , il sera pris égal a 10.

On peut aussi préciser que si le logiciel n'effacera pas le texte en mode AUTO. Il peut arriver qu'une ligne déja existante soit affichée , alors les lignes du texte sont décalées de 1 vers le bas.

SAUVER LE TEXTE.

S,Nom sauvera sur disquette le fichier de texte.

CHARGER UN TEXTE.

Cette fonction est équivalente au MERGE DU Basic. Si un texte est déja présent en mémoire , le fichier chargé sera mis à la suite de celui en mémoire (les numéros des lignes du texte seront reinitialisés). Si aucun texte n'est présent en mémoire , il sera simplement chargé. La syntaxe de cette fonction est la suivante.

G,Nom.

Les messages d'erreurs de l'éditeur de texte.

* Erreur de syntaxe : ce message apparaitra lorsque vous tenterez d'effectuer une commande inexistante , ou lorsque le nombre de données en entrée d'une commande n'est pas correcte.

* Ligne inconnue : apparaitra si vous demandez à l'éditeur de travailler avec une ligne qui n'existe pas.

* No de ligne incorrect : lors de certaines opérations , il se peut que les numéros de lignes dépassent les limites admises ; dans ce cas , ce message est lancé et le texte subit une renumérotation automatique.

* Buffer plein : le texte a atteint sa longueur maximale.

* Label inconnu : lorsque l'on recherche une ligne avec un label qui n'existe pas dans le texte.

* Mauvais fichier : survjendra lorsque l'utilisateur tentera de charger dans le buffer de texte , un fichier qui n'a pas été créé par cet éditeur.

* Fichier trop long : la jonction de deux fichiers sources peut dépasser la limite permise.

* Label incorrect : la taille maximum d'un label est de huit caractères.

* Mnémonique incorrect : le mnémonique de la ligne en question n'a pas été reconnu.

l'ASSEMBLEUR.

Ce programme transformera le programme source effectué avec l'éditeur de texte en un code objet éxecutable par l'AMSTRAD. Les directives de l'assembleur sont les suivantes :

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

La directive ORG.

Placé dans un programme , ORG fixera l'adresse ou sera assemblé le source. L'adresse transmise avec ORG ne devra correspondre à une zone libre de la RAM. Les adresses renvoyant à une zone ou est situé GT-Assembleur ou le programme source ne , seront pas acceptées. L'assembleur pourra par ailleurs assembler n'importe ou en RAM (dans les vecteurs systèmes par exemple) ou dans la mémoire écran ce qui fait que le source pourra prendre toute la zone libre (de &2500 à &A000) et le programmeur assemblera le texte dans la RAM ecran , en &C000. Il faut par conséquent prendre toute les précautions utiles (sauvegarde du texte...) avant d'assembler dans les vecteurs de l'AMSDOS car il y a un risque de plantage du système.

```
Exemple:    10          ORG &4000       ;origine en &4000
            20          LD A,2          ;
            30          JP &BC0E        ;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 etre placée juste après ORG. Placée avant ORG , elle sera annulée et placée plus loin dans le programme celui-ci n'aurait plus aucun sens.
LOAD fixera l'adresse ou le code objet sera chargé et éxecuté.

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

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

La directive ENT.

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

La directive EQU.

EQU initialise un label (chaine de 8 caractères maximum) 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 caractères 'label' , il prendra en compte l'entier &1234.

```
Exemple:  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 signifie Definition Byte et Definition Word. Leurs opérandes sont multiples :

```
        DB   "chaine de caractères"
ou      DW   "chaine de caractères"
```

pokera à la suite les caractères ASCII correspondants.

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

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

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

On peut aligner toutes ces operandes en les séparant par des virgules.

```
Ex:   DB  &12,100,"A"+10,label
```

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

Les messages d'erreurs de l'assembleur.

* Erreur de syntaxe : la ligne à assemblercomporte une erreur d'écriture dans les opérandes ou sa syntaxe n'est pas permise en assembleur Z80.

* Deplacement incorrect : dans les instructions de saut relatif (JR, DJNZ , JRcond.) , certaines limites ne doivent pas etre dépassées.

* Mauvaise origine : l'assembleur ne peut pas assembler sur lui-meme ou sur le texte (dans ce cas , utilisez la commande LOAD).

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

* Labels similaires aux lignes _ et _ : l'assembleur vérifiera que vous n'avez pas entré deux fois le meme label.

LE MONITEUR.

Il a les possibilités suivantes:

FONCTION	COMMANDE
Désassembler	D,adresse
Exécuter pas à pas	T,adresse
Interprétation du 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,longueur,valeur
Transformer du code en source	N,adr1,adr2,adr buffer
Brancher une ROM.	O,(A=amsdos) (D=dos) (B=basic) (E=externe)
Poker	P,adresse,octet
Voir et réinitialiser les ROMS	A
Sauver du code objet	S,nom,adr1,adr2,entrée
Charger un programme	G,nom,adresse
Reloger le moniteur	L,adresse
Exécuter un programme avec point d'arret.	R,adr,adr d'arret.

Passons maintenant en revue chacune de ces fonctions.

DESASSEMBLER.

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

D,adresse (entier entre 0 et 65535 : 0 et &FFFF).

Entre 0 et &3FFF , on pourra désassembler soit la RAM , soit la ROM.
Entre &C000 et &FFFF , 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 à la fois sur l'écran et sur l'imprimante.
On quitte le désassembleur par <ESCape>.

EXECUTION PAS A PAS.

Cette commande (T,adresse) provoque l'exécution 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 <escape>.

INTERPRETATION DU CODE OBJET.

I,adresse provoque l'exécution du code objet à l'adresse en entrée.
Cette exécution est ralentie par l'interprétation et peut-etre arretée par la touche <escape>.
Il faut noter que l'interpréteur ne vérifie rien ; ceci pourra mener à des plantages du système si par exemple il exécute son auto-destruction (réinitialisation de la mémoire par exemple). En outre ,l'interpréteur utilisant certaines routines de l'AMSDOS , il peut arriver qu'il entre en conflit avec lui-meme s'il interprete ces memes routines : le programme s'arrete alors. Enfin , l'interprétation des entrées-sorties est évidemment impossible ; certains messages d'erreur s'affichent et il vaut mieux sortir par <escape>.

VUE GLOBALE DE LA MEMOIRE.

Cette commande (M,adresse) donnera un apercu complet de la mémoire avec la valeur hexadécimale des octets et la valeur des caractères 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,adresse1,adresse2,longueur réalisera cette fonction. Il faut préciser que le moniteur ne vérifie rien , on pourra planter le système. Le copie de zones de mémoire pourra se faire partout sauf sur le moniteur.

REMPLISSAGE D'UNE ZONE DE MEMOIRE.

La commande F,adresse,longueur,octet remplira la zone entre 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 etre utile de mettre une zone de mémoire sous forme de texte source. Cette commande mettra sous de texte source , le texte du désassembleur :

N,adresse1,adresse2,buffer.

Adresse1 et Adresse2 représentent l'adresse de départ et d'arrivée du code à transformer. 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 de 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 :

O,A branche l'amsdos.
Un désassemblage à l'adresse 0 donne le décodage de la ROM.
O,B branche le BASIC.
Un désassemblage à l'adresse &C000 donne le listing du BASIC.
O,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 de n'importe quelle touche provoque la réinitalisation , c'est à dire le brachement 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 vérifiera s'il ne se détruit pas en chargeant le programme ; les mauvaises adresses seront rejetées.

RELOGER LE MONITEUR.

Le moniteur est entièrement 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 donc 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 complètement 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 à &4005 , il y aura un retour dans le mode pas à pas du moniteur.

R, adresse de départ, point 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 etre multiples :

- erreur de frappe ex: &963A
- nombre d'opérandes incorrect ex: d,100,1

* Adresse incorrecte : le moniteur ne peut pas se détruire en "polant" sur lui-meme (par exemple).

ANNEXE 1.

Caracteristoues des fichiers de GT-Assembleur.

ADES6128.BAS : lanceur BASIC du logiciel pour CPC 6128.

ADES664.BAS : lanceur BASIC du logiciel pour CPC 664.

ADES6128.BIN : noyau du logiciel pour CPC 6128.

ADES664.BIN : noyau du logiciel pour CPC 664.

GTEDIT.BIN : éditeur de texte.

Adresse de depart: &420
Longueur : &1510
Point d'entrée : &420

GTASSM.BIN : assembleur.

Adresse de depart: &420
Longueur : &1600
Point d'entree : &420

GTDESA.BIN : moniteur.

Adresse de depart: &420
Longueur : &2090
Point d'entree : &420

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

ANNEXE 2.

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

&BD3A pour un cpc 464.

&BD5B pour un cpc 664.

&BD5E pour un cpc 6128.

Le logiciel comporte deux lanceurs : ADES6128.BAS et ADES664.BAS. Le programme pourra etre lance sur CPC 6128 par l'un ou l'autre de ces deux lanceurs.

Par contre , sur un CPC 664 ou 464 , le logiciel ne pourra etre lance que par ADES664.BAS. Auparavant , il faudra modifier les trois modules :

GTEDIT.BIN
GTASSM.BIN
GTDESA.BIN

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

On fera le petit programme suivant :

```
10  memory &3fff
20  input "cpc 464 (1) ou 664 (2)";cpc
30  if cpc=1 then addfai=&3A 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 &40b0,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 PROGICIEL NE PEUT FONCTIONNER QUE SUR 6128

Cet utilitaire s'adresse aux personnes utilisant le langage machine du Z 80, 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 une trentaine de commandes qui en font un utilitaire relativement complet. Il permet entre autre, une utilisation très approfondie du lecteur de disquettes.

DEBUGG permet de pouvoir suivre le fonctionnement de n'importe quel programme en langage machine grâce à certaines de ses particularités:
- Il n'occupe aucune place en mémoire, étant entièrement résident dans les 64 K. suppléméntaires du 6128. Ceci permet donc d'utiliser les 64 K. courants comme on l'entend, sans problème de superposition.
- Il est autonome. DEBUGG n'utilise les routines systèmes que pour la gestion des disquettes. On peut donc utiliser les parties de la RAM réservées au sytème sans altérer son fonctionnement. Ce n'est que lors d'un accès aux disques qu'il est necessaire que la RAM système soit intacte.
- Il peut exécuter pas à pas toutes les instructions du Z 80 Il connait méme celles que l'on ne trouve dans aucun livre, qui utilisent par exemple des demi-registres d'index IX ou IY (XH,XL,YH ou YL). Ou bien encore,les codes CB 30 à CB 37 qui correspondent à une instruction de décalage inconnue que j'ai baptisée SLL r :
C <- registre r <- 1
- Il conserve en mémoire l'écran de travail. Cet écran esr réaffiché chaque fois qu'une commande est suceptible de le modifier ou alors, ces modifications sont redirigées vers la mémoire tampon. De méme, toutes les modifications de couleur ou de structure.

VIGNOUD Stéphane.

LES COMMANDES DE DEBUGG

Certaines commandes doivent être suives de paramètres. Si ceux-ci sont incorrects, DEBUGG les redemande un par un, en spécifiant à chaque fois la nature du paramètre demandé.
Les valeurs qui sont éventuellement poposées dans la fenêtre de droite sont celles prises par défaut.
Une valeur peut être rentrée sous forme hexadécimale, elle doit être alors suivie de la lettre H. (ex: BB06H).
Un paramètre peut être le résultat d'une opération; dans ce cas, les priorités de signe ne sont pas respectées.

Dans la liste qui suit, les paramètres mis entre parenthèses sont facultatifs.

RETURN — Visualise l'écran de Travail.

↑ ,↓ ,→ et ← — 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 à add, de longueur long et l'octet oct . L'adresse de début par défaut est la prmière 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 premiére adresse de la fenêtre. Ensuite :
↓ et ↑ : déplace le pointeur.
ESPACE : place un Break à l'adresse pointée. C'est à dire que lorsqu'on lance une exécution, celle ci est stoppée par un Break. Pour enlever un Break, il faut appuyer de nouveau sur ESPACE.
RETURN : ramène l'adresse pointée en haut de la fenêtre.
Q : sort du mode Désassemblage.

E(add),(break1,break2,..) Exécute un programme à partir de add ou par défaut, de l'adresse contenue dans le registre PC. Les adresse suivantes sont des adresses suivantes sont des adresses à partir desquelles l'exécution est stoppée.

F(add),(oct1,oct2,..) Recherche la chaine d'octets oct1,oct2,... en mémoire, à partir de add. Si la chaine d'octets n'est pas specifiée, c'est la précedente 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 imprimante de la zone mémoire allant de deb à fin. Si l'option DESA est selectionné, la sortie se fera sous la forme d'un désassemblage, sinon ce sera les codes et les caractères ASCII qui seront imprimés.

J Execute une instructon du type CALL, CALL Z, ... 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 les caractères 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 indiquee. L'adresse de début par défaut est la première de la fenêtre.

O(add),long,octet Applique un OU logique (OR).

P(add),(nb) Exécute en pas à pas nb instructions du Z 80 (par défaut une), à partir de add ou de l'adresse contenue 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 Sauvegarde un fichier de type binaire (2).

Tsource,long,dest Déplace la partie de la mémoire commençant à source et de longueur long jusqu'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 affiché.

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 Apllique un OU exclusif (XOR) .

Z Efface un fichier sur le lecteur courant.

DI DEBUGG interdira les interruptions lorsqu'il lancera une exécution (commandes E et J) 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 premier 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 un 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 registre, en mettant un R puis le nom du registre.
Exemple: RHL, RIX ou RPC