

# **ADEBUG**

**Brainstorm © 1990-1995** 

# Mode d'emploi

La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part que les "copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective", et, d'autre part, que les analyses et courtes citations dans un but d'exemple et d'illustration, "toute représentation ou reproduction intégrale ou partielle, faite sans le consentement des auteurs ou de leurs ayants droit ou ayants cause, est illicite" (alinéa 1 de l'article 40). Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait une contrefaçon sanctionnée par les articles 425 et suivants du Code Pénal.

Les marques citées sont des marques déposées par leurs propriétaires respectifs.

Brainstorm n'offre aucune garantie, expresse ou tacite, concernant ce logiciel et ne pourra être tenu responsable des préjudices éventuels dus à son utilisation.

### **Avant-propos**

Ce manuel d'utilisation a été conçu pour vous fournir toute l'information nécessaire à l'apprentissage et l'utilisation de votre débogueur Adebug.

Ce guide se divise en une introduction, deux parties principales et un ensemble d'appendices.

L'introduction présente ce qu'est un débogueur, et les connaissances qu'il nécessite.

La première partie contient une description rapide des différentes possibilités d'Adebug.

La seconde partie contient l'explication de toutes les fonctions d'Adebug, classées par ordre thématique.

Les appendices permettent d'accéder plus rapidement et plus précisément à divers éléments.

Adebug a été conçu avec le plus grand soin, et est utilisé par de nombreuses personnes depuis plusieurs années. Néanmoins, nous ne pouvons vous assurer de son parfait fonctionnement dans certaines conditions bien particulières. Nous vous remercions d'avance de bien vouloir nous signaler par courrier vos griefs, remarques, conseils, ... ou compliments.

#### ✓ Remerciements

Nous tenons à remercier toutes les personnes qui nous ont aidés et soutenus pour la maintenance et l'amélioration d'Adebug:

Gilles Berlioux, Xavier Cany, Emmanuel Collignon, Yovan Matovic, Pascal de France.

# Table des matières

Avant-propos	
1 Introduction	
1.1 Contenu de la disquette	
1.2 Qu'est-ce qu'un débogueur	
1.3 Qu'est-ce qu'un débogueur symbolique	
1.4 Connaissances nécessaires	
2 Description rapide	10
2.1 Introduction	
2.2 Adebug, un programme n'utilisant pas GEM	
2.3 Saisie d'une commande - Ligne de commande	
2.4 Variables	
2.5 Macros	16
2.6 Configuration	
2.7 Ligne de commande à l'exécution	17
2.8 Usage d'un terminal	18
2.9 Exemple d'utilisation	19
3 Références	22
3.1 Liste thématique des fonctions	22
3.2 Liste alphabétique des fonctions	26
3.3 Fenêtres	29
[Alt_Z]oom de fenêtre	29
[Tab]	
[Alt_S]épare/Regroupe fenêtres	30
[Alt_T]ype de visualisation	31
[0~7] Changer de type la fenêtre courante	31
[Alt_1~5]	31
[Ctl_0~7]	31
[Ctl_8]	32
[Ctl_9]	32
[Ctl_Alt_1~5]	32
[Sft_Ctl_0~7]	32
[Sft_Ctl_8]	33
[Sft_Ctl_9]	33
[Flèche haut]	33
[Flèche bas]	33
[Flèche gauche]	33
[Flèche droite]	33
[Ctl_Flèche gauche]	33
[Ctl_Flèche droite]	34
[Sft_Flèche haut]	34
[Sft_Flèche bas]	34
[Ctl_Flèche gauche]	
[Ctl_Flèche droite]	
Ctl_Flèche haut]	
[Ctl_Flèche bas]	
-	

BRAINSTORM

[A]dresse de fenêtre	34
[Alt_L]	34
[W]atch	35
[Sft_Watch]	35
[#]	35
[Ctl_Home]	36
[Sft_Ctl_Home]	36
[Ctl_Alt_G]	36
[Ctl_Alt_S]	36
[Ctl_Alt_L]	36
[Ctl_I]	
3.4 Gestion et édition mémoire	
[Alt_E]dition de fenêtre	37
[C]opier	41
[F]ill	41
[G]et expression	42
[N]ext	42
[Aİt_N]	42
3.5 Macros	
[M]acro	44
3.6 Variables	
[Alt_V]ariable	48
adebug.var	
[Ctl_Alt_V] Charger les symboles	
[Ct1_V]	
[Help]	
[L]iste des variables	
3.7 Écran	
[V]oir	
[Ctl_Alt_I]nverse	
3.8 Disque	
[D]irectory	
[Alt_D]irectory	
[Sft_Alt_D]irectory	
[Ctl_L]oad program	
[B]inary load	
[Alt_A]SCII	
[S]ave binary	
3.9 Configuration	
[Ctl_P]références	
3.10 Opérations de contrôle de flux du programme	
[Sft_Ctl_C]	
[Ctl_Z]	
[Ctl_S]kip	
[Ctl_R]un	
[Ctl_A]rrêt	68

[T]race (Jusqu'à, Instruction, Rien, 68020)	
[U]ntil	
[Ctl_U]ntil	
[J]ump	
[Ctl_J]ump	. 70
[Ctl_T]race	, 70
[Ctl_F]orce	
[Ctl_eX]écute	. 70
[Alt_eX]écute	. 70
[Ctl_Alt_J]sr to subroutine	. 70
[Ctl_Alt_Z] Trace T1	. 71
[Sft_Ctl_Alt_J]sr to subroutine	. 71
[Sft_Alt_Help]	. 71
[Ctl_Alt_F]	. 72
3.11 Points d'arrêt	
[Ctl_B]reakpoint	. 73
[Alt_B]reakpoint	. 74
[Ctl_K]ill	
Ctl_E xception	
[Ctl_D]étourne	
[Ctl_Alt_D]	
[Ctl_Alt_B]	
3.12 Commandes diverses	
[K]eep	
[R]estore	
[I]nterrupt Priority Level	
[Alt_Help]Impression d'écran	
[Alt_M]emory free	
[H]istorique	
[F2]	
[F5]	
[Alt_P]rint	
[P]rint	
[E]value	
[Alt_@]	
[Ctl_Alt_R]	
[O]utput	
3.13 Profiler (échantillonneur de temps d'exécution)	
[Alt_R]emettre à zéro le profiler	
APPENDICES	
Appendice I - Connaissances nécessaires à l'usage d'un	, 00
débogueur	. 83
Le microprocesseur	
L'assembleur	
Les modes d'adressage	
Les instructions du 68xxx	
Notions de trace	

Notions de point d'arrêt / exception	96
Notions d'interruption / exception	
Gestion d'une interruption par le 68xxx	98
Prise en charge d'une interruption par le 68xxx	
Priorité des interruptions	
Appendice II - Périphériques utilisables	
1 - Ecran	
2 - Disque (souple et dur)	103
3 - Le port série (RS232)	
4 - Le port parallèle (imprimante)	
Appendice III - L'évaluateur	106
Appendice IV - Liste thématique des messages d'erreur	
Appendice V - Petite documentation de l'Atari	
Appendice VI - Liste des figures	136
Appendice VII - Table ASCII	
Appendice VIII - Index général	138

# 1 Introduction

#### 1.1 Contenu de la disquette

Adebug n'est pas protégé contre la copie. Nous vous conseillons donc fortement de faire une copie de travail de la disquette originale, puis de ranger celle-ci.

Vous trouverez à votre disposition les fichiers suivants:

LISEZMOI.TXT un fichier ASCII contenant des ajouts à ce

manuel.

**adebug.prg** Le programme lui-même.

**rdebug.prg** La version résidente du débogueur.

**adebug.var** fichier de variables.

**a\_debug.mac** fichier de macro par défaut.

exemple.prg un programme d'exemple de débogage.

exemple.s le source de ce programme.

Un répertoire ros contenant les routines (\*.ro).

Un répertoire rosrc contenant les sources des routines.

### 1.2 Qu'est-ce qu'un débogueur

On pourrait donner comme définition d'un débogueur: "outil d'aide à la recherche et à la correction des erreurs d'analyse et de programmation". Cependant, en informatique, comme partout ailleurs, on n'a "rien sans rien"; un débogueur ne peut travailler que sur un **programme exécutable**, c'est à dire directement interprétable par le microprocesseur. et par conséquent difficilement compréhensible pour le programmeur. Et puisqu'il faudra bien en arriver là, disons-le: pour utiliser un débogueur, une bonne connaissance de l'assembleur est indispensable. Mais rassurez-vous, si vous ignorez encore tout de ce langage, cela ne saurait durer après la lecture de la partie correspondante dans l'appendice 1 Connaissances nécessaires à l'usage d'un débogueur.

En règle générale, on utilise un débogueur pour rechercher l'endroit où un programme "plante", puis pour essayer de comprendre pourquoi tout ne fonctionne pas comme prévu. Si cette première étape s'est passée de façon correcte, et c'est à **cela** que sert un débogueur, il devient bien plus simple de corriger les instructions fautives du programme source, voire même de modifier la logique du programme si l'erreur est plus grave, et ce quel que soit le langage utilisé... Surtout il ne faut pas croire que le débogueur est l'outil réservé du programmeur en assembleur.

En effet, Adebug permet de suivre pas à pas l'exécution de n'importe quel programme, et ceci même s'il n'a pas été écrit en assembleur. Il n'est en fait limité à aucun langage de programmation et, même si une bonne connaissance de l'assembleur est requise, il vous permettra grâce à sa gestion des **symboles** de déboguer n'importe quel programme en quelque langage qu'il ait été écrit.

### 1.3 Qu'est-ce qu'un débogueur symbolique

Lorsqu'on lit un programme écrit dans un langage dit 'évolué', on trouve ce qu'on appelle des variables. Elles simplifient la compréhension du programme en donnant un nom à des objets logiques. Il est ainsi plus simple de comprendre un GOSUB lecture dans un programme écrit en basic qu'un JSR \$FC098C dans un programme chargé en mémoire avec Adebug. Il vous faut cependant savoir que le format d'un programme exécutable sur Atari comprend une section qui contient tous les noms de variables et de sous-programmes utilisés lors de l'écriture du programme. Adebug sait relire cette section et, si votre langage le permet, il vous sera possible de visualiser ces noms pour simplifier le débogage. Ainsi le GOSUB lecture du basic sera affiché sous la forme JSR lecture par Adebug... et à côté de l'adresse \$FC098C vous trouverez le label lecture! En outre, comme Adebug connaît (grâce au fichier adebug.var) les fonctions du système de l'Atari, vous aurez parfois l'impression en traçant votre programme (voire la ROM) d'en lire le listing source!

#### 1.4 Connaissances nécessaires

Pour comprendre parfaitement ce manuel, quelques notions sur des sujets fondamentaux sont indispensables:

l'assembleur, le mode trace, les points d'arrêt, les exceptions, les périphériques.

Pour une initiation ou un approfondissement sur l'un de ces thèmes, lisez la partie correspondante dans l'appendice 1 Connaissances nécessaires à l'usage d'un débogueur.

# 2 Description rapide

#### 2.1 Introduction

#### **Notations**

Dans toute la suite de ce manuel, on indiquera:

les exemples avec le symbole les remarques simples avec le symbole les remarques importantes avec le symbole les remarques critiques avec le symbole les remarques de la complex de la com

On notera entre crochets ("[]") les combinaisons de touches en utilisant les conventions suivantes:

Ctl signifie touche **Control** enfoncée. Alt signifie touche **Alternate** enfoncée. Sft signifie touche **Shift** enfoncée.

Une lettre seule signifie la **touche correspondant à cette lettre** sur le clavier enfoncée.

[Ctl\_Z]

signifie touche Control et touche z pressées simultanément.

Un chiffre seul signifie la **touche correspondant à ce chiffre** sur le clavier (pavé numérique et rangée supérieure du clavier sans [Sft]) enfoncée.

[Ctl\_2]

signifie touche Control et touche 2 pressées simultanément.

Une lettre F suivie d'un numéro symbolise la **touche de fonction** correspondant à ce dernier.

[F1]

signifie touche de fonction 1 enfoncée.

#### Clavier

- L'état de la touche **[Caps Lock]** n'influe pas sur la prise en compte des raccourcis clavier.
- Lorsque le niveau d'IPL interne d'Adebug (cf. la commande [I]) est supérieur à 5, la gestion clavier d'Adebug est totalement déconnectée de celle du système.

En conséquence, lorsqu'une touche est enfoncée sous Adebug et relâchée en dehors de celui-ci, le relâchement ne peut être pris en compte par le débogueur, ce qui peut être gênant, notamment pour les touches mortes (Sft, Alt, Ctrl, etc.). L'appui sur **[F5]** n'importe quand sous Adebug permet de remettre à l'état "relâché" toutes les touches du clavier.

### 2.2 Adebug, un programme n'utilisant pas GEM

Pour des raisons de rapidité d'utilisation et d'affichage, Adebug ne fait pas appel aux fonctions de l'interface GEM. En effet, bien que l'usage exclusif de **raccourcis clavier** impose à l'utilisateur un certain temps d'apprentissage, cette méthode permet de déboguer de la façon la plus rapide et la plus aisée. En outre, une quelconque dépendance vis à vis d'un système d'exploitation n'est jamais bonne pour un débogueur; et, à la différence de nombre de ses pairs, si jamais le système d'exploitation "plante", Adebug, lui, ne "plante" pas!

Mais cette absence totale d'utilisation des fonctions du système n'exclut pas l'usage d'un **multifenêtrage**, qui demeure l'une des meilleures interfaces utilisateur. C'est pour cette raison qu'Adebug utilise des fenêtres pour afficher ses données.

Cinq fenêtres peuvent être affichées simultanément, en sept types de visualisation:

registres du 68xxx désassemblage 'dump' hexadécimal ASCIIsource (en mode source) variables (en mode source) inspect (en mode source)

Toutes les fenêtres peuvent adopter n'importe lequel de ces types. Seule la fenêtre 2 possède un comportement différent des autres. A chaque changement de contexte (trace, point d'arrêt, etc....), Adebug vérifie que le PC ("Program Counter", ou compteur ordinal) se trouve à l'intérieur. Si tel n'est pas le cas, l'adresse de base de la fenêtre 2 se trouve forcée à la valeur du PC.

En type désassemblage:

Les noms des appels Ligne A sont affichés à droite des instructions.

Les noms des appels Gemdos, Bios, Xbios, Aes et Vdi sont affichés à l'exécution du passage de paramètres.

La **fenêtre active** ou **fenêtre courante** est représentée avec un cadre plus épais.

Les fonctions agissant sur une fenêtre le font toujours sur la fenêtre courante.

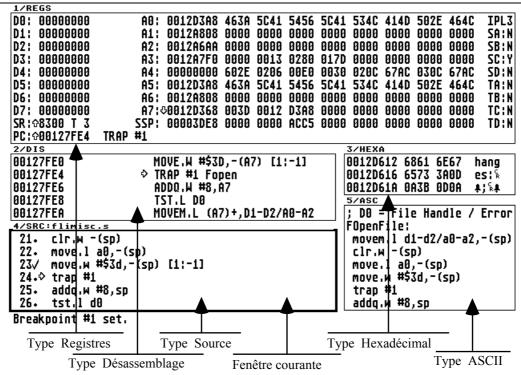


Figure 1: les cinq fenêtres d'Adebug en cinq types de visualisation.

### 2.3 Saisie d'une commande - Ligne de commande

La **ligne de commande** sert à entrer les paramètres des diverses fonctions d'Adebug qui le nécessitent.

Une fois le ou les (dans ce cas, séparés par une virgule) paramètres entrés, il faut taper [Return] (qui tronque la ligne au niveau du curseur) ou [Enter] (qui saisit toute la ligne indépendamment de la position du curseur) afin de valider la commande. Pour effacer un caractère situé avant le curseur, il suffit de taper [Backspace]; s'il se trouve sous le curseur, il faut utiliser [Delete].

Les flèches gauche et droite du pavé intermédiaire, quant à elles, servent à déplacer le curseur sur la ligne (les flèches haut et bas servent à l'historique, développé ci-dessous). Si une des touches [Sft] est pressée simultanément avec [Flèche gauche] (respectivement [Flèche droite]), le curseur se positionnera en début (respectivement en fin) de ligne.

Autres touches disponibles dans la ligne de commande:

[Esc] permet de sortir de la ligne de commande.

[Ctl\_Del] efface la totalité de la ligne.

[Ctl\_Tab]réalise la complétion de la variable se trouvant sous le curseur.

**[Undo]** restaure la ligne intégralement après toute série de délétions.

[Ctl\_C]ase inverse majuscule/minuscule sous le curseur.

[Ct1\_L]ower transforme tous les caractères alphabétiques majuscules présents sur la ligne en leur minuscule.

[Ctl\_M]atching réalise la correspondance des parenthèses, crochets et accolades.

**[Ct1\_R]** prend le code ASCII (en hexa) d'un caractère et affiche ce dernier.

Les touches n'appartenant pas à {0..9}U{A..F} sont directement interprétées.

[Ct1\_T]widdle inverse les deux caractères situés à gauche du curseur.

[Ctl\_U]pper transforme tous les caractères alphabétiques minuscules présents sur la ligne en leur majuscule.

# Les raccourcis clavier compatibles Tcsh sont également présents:

[Ctl\_G]=[Esc] [Ctl\_H]=[Backspace] [Ctl\_P]=[Up]

 $[Ctl_N]=[Down]$ 

[Ctl\_B]=[Left]

[Ctl\_F]=[Right]

[Ctl\_A]=[Sft\_Left]

[Ctl\_E]=[Sft\_Right]

La ligne de commande peut contenir 200 caractères au maximum.

#### Historique

Les flèches haut et bas servent respectivement à monter et à descendre dans l'**historique** des commandes. Ce dernier est constitué des précédentes entrées de la ligne de commande à concurrence de la taille du tampon d'historique (nommé **HIS** dans les préférences) fixée. S'il n'y a plus de commande à afficher, l'écran clignote en signe d'erreur.

#### 2.4 Variables

Une autre des particularités d'Adebug est de pouvoir gérer différents types de **variables**, propres au programme débogué ou créées par l'utilisateur Ainsi, par exemple, le type de variable nommé **RO** permet d'inclure des routines supplémentaires dans Adebug afin que l'utilisateur puisse disposer de nouvelles fonctions, spécifiques à son application, et écrites par lui. Cette modularité d'Adebug permet à chacun de disposer de son propre débogueur adapté à ses besoins et exigences.

Les différents types de variables sont décrits en détail dans le chapitre de référence sur les variables.

#### 2.5 Macros

#### Utilisation

Le but de l'utilisation des **macros** est d'éviter de refaire à la main plusieurs fois les mêmes manipulations. Il ne s'agit pas d'un réel langage de commande, même si certaines fonctionnalités s'en rapprochent.

#### **Syntaxe**

Une macro est avant tout une représentation **symbolique**des fonctions d'Adebug.

Chaque touche associée à une fonction est représentée par:

- Un signe ` (backquote).
- L'état des touches: sft\_, alt\_, ctl\_.
- La lettre ou le chiffre correspondant à la touche du clavier utilisée, ou un mot la symbolisant.

Une macro est stockée sous forme d'un fichier ASCII. Il est donc possible de créer ou de modifier une macro sous un quelconque éditeur de texte. Mais il est souvent plus simple d'enregistrer la séquence de manipulation grâce à la fonction [M]acro **E)nregistrement**.

#### 2.6 Configuration

Afin de ne pas être obligé de le reconfigurer après chaque lancement, Adebug permet de sauvegarder sous la forme d'un fichier (**adebug.sav**) un certain nombre de paramètres et d'options de débogage.

L'explication complète des données configurables figure dans le chapitre de référence sur la configuration.

### 2.7 Ligne de commande à l'exécution

Sous shell ou depuis le bureau, il est possible de transmettre une **ligne de commande** à Adebug. Cette ligne contient:

Soit le nom d'un programme à charger (elle est alors équivalente à la fonction [Ctl\_L], voir le chapitre de référence sur les fonctions disque), suivi de la ligne de commande éventuelle du programme.

#### adebug exemple

Lance Adebug en lui demandant de charger le programme exemple.

#### adebug exemple paramètre

Lance Adebug en lui demandant de charger le programme exemple en lui passant paramètre comme ligne de commande.

Soit l'option -b suivie d'un nom de fichier binaire à charger (équivalente à la fonction [Ctl\_B], voir le chapitre de référence sur les fonctions disque).

#### Au bureau:

-b data.bin dans la fenêtre de saisie des paramètres.

#### Sous shell:

adebug -b data.bin

Lance Adebug en lui demandant de charger le fichier binaire data.bin.

### 2.8 Usage d'un terminal

Il est pratique de pouvoir visualiser à la fois l'écran d'Adebug et celui du programme en cours de débogage. De plus, certains programmes peuvent utiliser l'écran de l'Atari en adressant les circuits vidéo directement de telle façon qu'il est impossible à Adebug de conserver un affichage propre. Pour ces raisons, il est prévu de pouvoir se servir pour l'affichage d'un terminal déporté. Cette fonction ([O]) permet d'utiliser un terminal (Minitel, par exemple) relié à l'Atari par la prise série. Tous les affichages se feront dès lors sur ce terminal et l'écran de l'Atari sera entièrement laissé au programme en cours de débogage. Le Minitel 2 est probablement le meilleur choix pour cette utilisation. Il permet en effet une liaison série à 9600 bauds (soit deux fois plus rapide que les anciens Minitels) qui est suffisante pour une utilisation agréable. Bien sûr, si vous disposez d'un terminal professionnel, il vous sera possible d'augmenter la vitesse de communication jusqu'à 19200 bauds.

La liaison physique entre l'Atari et le terminal doit se faire par un câble adapté. Pour un Minitel, les câbles sont en vente partout. Pour un autre terminal, utilisez un câble 'null-modem'.

► Seul l'affichage se fait sur le terminal déporté. La saisie des commandes se fait toujours sur le clavier de l'Atari, même dans ce mode.

#### 2.9 Exemple d'utilisation

Et maintenant, un petit exemple d'utilisation. Lancez Adebug, puis à la demande:

#### Charger prg:

entrez le nom de programme: exemple

Le message:

Fichier <exemple.prg> en cours de chargement. apparaît, puis un deuxième message:

Fichier <exemple.prg> chargé en 96 octets.

Les fenêtres 2 et 3 affichent le début du programme chargé. Dans la première, en haut et à gauche, un symbole:

#### TEXT

est affiché. Il représente le début du programme. Entre ce symbole et l'instruction, une petite flèche indique l'adresse contenue dans le **PC** ("**Program Counter**", ou compteur ordinal). C'est l'instruction par laquelle commence le programme.

Après avoir appuyé sur [Ctl\_Z], le message:

Traced. (tracé)

apparaît. Cela signifie que vous avez tracé une instruction.

De plus, dans la fenêtre 1, le contenu du registre D0 a changé, comme l'indique la petite flèche à gauche du registre (flèche vers le haut si le registre a été incrémenté, et vers le bas s'il a été décrémenté). Il vaut maintenant:

1234

soit \$1234 (en hexadécimal).

La fenêtre 2 a été aussi modifiée. La flèche est descendue d'une ligne, l'instruction qu'elle indiquait précédemment ayant été exécutée.

En recommençant la même opération, une nouvelle instruction est exécutée, et la flèche est parallèlement descendue. Mais c'est maintenant une flèche vers le bas. En effet, la prochaine instruction est une instruction de branchement (BSR) à une adresse supérieure à l'actuel PC.

De nouveau grâce à **[Ctl\_Z]**, le branchement est effectué. La flèche indique de nouveau la droite, ce qui signifie que la prochaine instruction ne fera pas de branchement. Dans la fenêtre de registres, seule la pile (registre A7) a été modifiée (ainsi que le SR et le PC bien sûr). La première ligne de la fenêtre 2 indique le PC,

car ce dernier doit toujours s'y trouver.

Après avoir tracé la prochaine instruction (avec **[Ct1\_Z]**), rien n'a changé dans la fenêtre 1 (car l'instruction est un NOP), mais la flèche pointe maintenant vers le haut. En effet, l'instruction est un DBF, qui boucle sur le registre d0 vers le haut.

Le DBF branche sur le NOP immédiatement au-dessus (routin\_1). Il est donc inutile de continuer à tracer par **[Ct1\_Z]**. En faisant **[Ct1\_A]**, un point d'arrêt est mis sur l'instruction suivante, et le programme est lancé.

L'exécution est stoppée par le point d'arrêt, et le message **Run and break. (Lancé et stoppé.)** 

est affiché. Il ne reste plus qu'à effectuer le RTS (avec **[Ct1\_Z]**) pour revenir de la routine.

Les deux MOVE vers registre peuvent être tracés, jusqu'au BSR. Ce dernier branche vers la même routine que précédemment. Il suffit donc de faire [Ct1\_A] pour le passer.

Le BSR suivant ne branche pas sur la même routine. Il suffit de tracer jusqu'à lui, puis de faire **[W]atch**. L'adresse de la fenêtre 2 est maintenant l'adresse sur laquelle pointait le BSR. Il est alors aisé de remarquer que cette routine est en réalité presque la même que précédemment. En faisant de nouveau **[W]atch**, la fenêtre 2 est remise à son ancienne adresse et par **[Ctl\_A]**, la routine est effectuée.

Les deux dernières instructions forment l'appel GEMDOS servant à terminer un programme. Après l'exécution de ces instructions, le programme tracé est donc terminé.

```
;exemple.s, exemple de programme à tracer pour Adebug.
                ; pour avoir les symboles de débogage.
     opt d
Start:
                #$1234,d0
     move.w
                #2,d1
     moveq
                routin 1
     bsr
                #$4321,d0
     move.w
                #-2,d1
     moveq
                routin 1
     bsr
                #-2,d0
     moveq
                #$4321,d1
     move.w
                routin_2
     bsr
                #0,d0
     moveq
     rts
routin_1:
     nop
     dbf
                d0,routin_1
     rts
routin 2:
     nop
     dbf
                d1,routin_2
     rts
     END
```

# 3 Références

### 3.1 Liste thématique des fonctions

#### **Fenêtres**

[Tab] changer de fenêtre courante.

[0~7] changer de type la fenêtre courante.

[Ctl\_0~7] mettre l'adresse de la fenêtre courante à A0~7.

[Ctl\_8] mettre l'adresse de la fenêtre courante à SSP.

[Ct1\_9] mettre l'adresse de la fenêtre courante à PC.

[Ctl\_Alt\_1~5] mettre l'adresse de la fenêtre courante à l'adresse de la fenêtre 1~5.

[Sft\_Ctl\_0~7] mettre l'adresse de la fenêtre courante à {A0~7}.

[Sft\_Ct1\_8] mettre l'adresse de la fenêtre courante à {SSP}.

[Sft\_Ct1\_9] mettre l'adresse de la fenêtre courante à {PC}.

[Flèche haut] monter d'une ligne dans la fenêtre courante.

[Flèche bas] descendre d'une ligne dans la fenêtre courante.

[Flèche gauche] enlever un à l'adresse de la fenêtre courante.

[Flèche droite] ajouter un à l'adresse de la fenêtre courante.

[Sft\_Flèche haut] monter d'une page dans la fenêtre courante.

[Sft\_Flèche bas] descendre d'une page dans la fenêtre courante.

[Ctl\_Flèche gauche] diminuer la fenêtre courante vers la gauche.

[Ctl\_Flèche droite] agrandir la fenêtre courante vers la droite.

[Ctl\_Flèche haut] changer la hauteur de la fenêtre courante.

[Ctl\_Flèche bas] changer la hauteur de la fenêtre courante.

[A]dresse modifier l'adresse de la fenêtre courante.

[Alt\_L]ier la fenêtre courante à une expression.

[Alt\_1~5] aller sur la fenêtre 1~5.

[Alt\_S]éparer et regrouper les fenêtres.

[Alt\_T]ype de visualisation dans une fenêtre.

[Alt\_Z]oom agrandir la taille de la fenêtre courante.

[W]atch visualiser le contenu de l'adresse de branchement.

**[Sft\_W]**atch mettre la fenêtre courante à l'adresse de branchement.

[Ctl\_Alt\_G] voir les globales (en mode source).

[Ctl\_Alt\_L] voir les locales (en mode source).

[Ctl\_Alt\_S] voir les statiques (en mode source).

[Ctl\_Home] aller en début de source ou de variables (en mode source)

[Sft\_Ctl\_Home] aller en fin de source ou de variables (en mode source).

[#] aller au numéro de ligne spécifié (en mode source).

#### Gestion et édition mémoire

[C]opier une partie de la mémoire.

[F]ill remplir une partie de la mémoire.

**[G]**et chercher une expression.

[N]ext chercher en avant l'expression.

[Alt\_E]dition de fenêtre.

[Alt\_N]ext chercher en arrière l'expression.

#### **Macros**

[M]acro menu des macros

#### **Variables**

adebug.var fichier d'initialisation des variables.

[Help] afficher la liste des tampons.

[L]iste afficher la liste des variables.

[Ctl\_V] charger un fichier de variables.

[Ct1\_I]inspection de variable (en mode source).

[Alt\_V]ariable créer variable.

[Ctl\_Alt\_V] charger les symboles d'un programme en mémoire.

#### Écran

[V]oir l'écran logique.

[Ctl\_Alt\_I]nverser la palette de couleurs d'Adebug.

#### Opérations disque

[D]irectory changer de répertoire courant.

[Alt\_D]irectory afficher le répertoire courant et son contenu.

[Sft\_Alt\_D]irectory idem avec choix des path et extension.

[Ctl\_L]oad charger un programme.

[B]inaire charger fichier binaire.

[Alt\_A]SCII charger fichier ASCII.

[S]auver en binaire une partie de la mémoire.

#### Configurations

[Ctl\_P]références.

#### Opérations de contrôle de flux du programme

[Ctl\_C] terminer Adebug ou le programme en cours de débogage.

[Sft\_Ctl\_C] idem précédé de l'appel AES wind\_new().

[Ctl\_Z] tracer une instruction.

[Ct1\_S]kip passer l'instruction courante sans l'exécuter.

[Ctl\_R]un lancer l'exécution.

**[Ctl\_A]**rrêt placer un point d'arrêt à l'instruction suivante et lancer.

[T]racer de manière paramètrable.

[U]ntil placer un point d'arrêt et lancer.

[Ctl\_U]ntil placer un point d'arrêt sur la fenêtre courante et lancer.

[J]ump changer le PC.

[Ctl\_J]ump mettre le PC à l'adresse de la fenêtre courante.

[Ctl\_T]racer en restant toujours au plus haut niveau.

[Ct1 F]orcer le branchement.

[Ctl\_eX]écuter routine.

[Alt\_eX]écuter retour.

[Ctl\_Alt\_J]sr exécuter une routine en mode utilisateur.

[Ctl\_Alt\_Z] tracer en mode T1 du 68020.

**[Sft\_Ct1\_Alt\_J]**sr exécuter une routine en mode superutilisateur.

**[Sft\_Alt\_Help]** arrêter le programme en cours.

[Ctl\_Alt\_F]rame voir la dernière stack frame.

#### Points d'arrêt

[Ctl\_B]reakpoint mettre un point d'arrêt sur la fenêtre courante.

[Alt\_B]reakpoint mettre un point d'arrêt configurable.

 $[Ct1\_D]$ étourner un TRAP

[Ctl\_E]xception installer toutes ou certaines exceptions.

[Ct1\_K]ill détruire tous les points d'arrêts présents.

[Ctl\_Alt\_B] liste des points d'arrêt.

[Ctl\_Alt\_D] Afficher la liste des appels reconnus par [Ctl\_D]

#### **Profiler**

[Alt\_R]emettre à zéro le profiler.

#### Commandes diverses

[E]valuer une expression.

[H]istorique des instructions.

[I]nterrupt Priority Level régler le niveau d'IPL.

[K]eep mémoriser la valeur courante de tous les registres.

[O]utput afficher sur écran ou par port série.

[P]rint imprimer le désassemblage d'une zone mémoire.

[R]estaurer la valeur de tous les registres.

[Alt\_Help] Imprimer l'écran.

[Alt\_M]émoire libre.

[Alt\_P]rint imprimer le contenu de la fenêtre courante.

[Alt\_@] Afficher le copyright.

[Ctl\_Alt\_D] Liste des appels AES et VDI.

[Ctl\_Alt\_R]edémarrer à partir d'une adresse absolue.

[Ctl\_Alt\_Del]ete provoquer un reset à chaud

 $[\mathbf{Sft}\_\mathbf{Ctl}\_\mathbf{Alt}\_\mathbf{Del}]$ ete provoquer un reset à froid

[F1] poser marque.

[F2] échanger curseur et marque.

[F5] relâcher les touches shifts.

# 3.2 Liste alphabétique des fonctions

[#] aller au numéro de ligne spécifié (en mode source).

[Tab] changer de fenêtre courante.

[Flèche haut] monter d'une ligne dans la fenêtre courante.

[Flèche bas] descendre d'une ligne dans la fenêtre courante.

[Flèche gauche] enlever un à l'adresse de la fenêtre courante.

[Flèche droite] ajouter un à l'adresse de la fenêtre courante.

[Sft\_Flèche haut] monter d'une page dans la fenêtre courante.

[Sft\_Flèche bas] descendre d'une page dans la fenêtre courante.

[Help] liste des tampons.

[F1] poser marque.

[F2] échanger curseur et marque.

[F5] relâcher les touches shifts.

[0~7] changer de type la fenêtre courante.

[A]dresse modifier l'adresse de la fenêtre courante.

[B]inary charger fichier binaire.

[C]opier une partie de la mémoire.

[D]irectory changer de répertoire courant.

[E]valuer une expression.

[F]ill remplir une partie de la mémoire.

[G]et chercher une expression.

[H]istorique des instructions.

[I]nterrupt Priority Level régler le niveau d'IPL.

[J]ump changer le PC.

[K]eep mémoriser la valeur courante de tous les registres.

[L]iste des variables.

[M]acro menu des macros.

[N]ext chercher en avant l'expression.

[O]utput afficher sur écran ou sur port série.

[P]rint imprimer le désassemblage d'une zone mémoire.

[R]estaurer la valeur de tous les registres.

[S]auver en binaire une partie de la mémoire.

[T]racer de manière paramètrable.

[U]ntil placer un point d'arrêt et lancer.

[V]oir l'écran logique.

[W]atch visualiser le contenu de l'adresse de branchement.

[Alt\_Help] imprimer l'écran.

[Alt\_@] afficher le copyright.

[Alt\_1~5] aller sur la fenêtre 1~5.

[Alt\_A]SCII charger fichier ASCII.

[Alt\_B]reakpoint mettre un point d'arrêt configurable.

[Alt\_D]irectory afficher le répertoire courant et son contenu.

[Alt\_E]dition de fenêtre.

[Alt\_L]ier la fenêtre courante à une expression.

[Alt\_M]émoire libre.

[Alt\_N]ext chercher en arrière l'expression.

[Alt\_P]rint imprimer le contenu de la fenêtre courante.

[Alt\_R]emettre à zéro le profiler.

[Alt\_S]éparer et regrouper les fenêtres.

[Alt\_T]ype de visualisation dans une fenêtre.

[Alt\_V]ariable créer variable.

[Alt\_eX]écuter retour.

[Alt\_Z]oom agrandir la taille de la fenêtre courante.

**[Ctl\_Home]** aller en début de source ou de variables (en mode source)

**[Sft\_Ctl\_Home]** aller en fin de source ou de variables (en mode source).

[Ctl\_0~7] mettre l'adresse de la fenêtre courante à A0~7.

[Ctl\_8] mettre l'adresse de la fenêtre courante à SSP.

[Ct1\_9] mettre l'adresse de la fenêtre courante à PC.

[Ctl\_Alt\_1~5] mettre l'adresse de la fenêtre courante à l'adresse de la fenêtre 1~5.

**[Sft\_Ctl\_0~7]** mettre l'adresse de la fenêtre courante à  $\{A0~7\}$ .

[Sft\_Ct1\_8] mettre l'adresse de la fenêtre courante à {SSP}.

**[Sft\_Ct1\_9]** mettre l'adresse de la fenêtre courante à {PC}.

[Ctl\_Flèche gauche] diminuer la fenêtre courante vers la gauche.

[Ctl\_Flèche droite] agrandir la fenêtre courante vers la droite.

[Ctl\_Flèche haut] augmenter la hauteur de la fenêtre courante.

[Ctl\_Flèche bas] diminuer la hauteur de la fenêtre courante.

[Ctl\_A]rrêt placer un point d'arrêt à l'instruction suivante et lancer.

[Ctl\_B]reakpoint mettre un point d'arrêt sur la fenêtre courante.

[Ctl\_C] terminer Adebug ou le programme en cours de débogage.

[Ctl\_D]étourner un TRAP

[Ctl\_E]xception installer toutes ou certaines exceptions.

[Ctl\_F]orcer le branchement.

[Ctl\_I]inspection de variable (en mode source).

[Ctl\_J]ump mettre le PC à l'adresse de la fenêtre courante.

[Ctl\_K]ill détruire tous les points d'arrêts présents.

[Ctl\_L]oad charger programme.

[Ctl\_P]références changer et sauver les préférences.

[Ctl\_R]un lancer l'exécution.

[Ct1\_S]kip passer l'instruction courante sans l'exécuter.

[Ctl\_T]racer en restant toujours au plus haut niveau.

[Ctl\_U]ntil placer un point d'arrêt sur la fenêtre courante et lancer.

[Ctl\_V]ariable charger un fichier de variables.

[Ctl\_eX]écuter une routine.

[Ctl\_Z] tracer une instruction.

**[Sft\_W]**atch mettre la fenêtre courante à l'adresse de branchement.

[Ctl\_Alt\_Del]ete provoquer un reset à chaud

**[Ctl\_Alt\_1~5]** donner à la fenêtre courante l'adresse de la fenêtre 1~5.

[Ctl\_Alt\_B] lister les points d'arrêt.

[Ctl\_Alt\_D] affiche la liste des fonctions reconnues par [Ctl\_D]

[Ctl\_Alt\_F]rame voir la dernière stack frame.

[Ctl\_Alt\_G] voir les globales (en mode source).

[Ctl\_Alt\_I]nverser la palette de couleurs d'Adebug.

[Ctl\_Alt\_J]sr exécuter une routine en mode utilisateur.

[Ctl\_Alt\_L] voir les locales (en mode source).

[Ctl\_Alt\_R]edémarrer à partir d'une adresse absolue.

[Ctl\_Alt\_S] voir les statiques (en mode source).

[Ctl\_Alt\_V] charger les symboles d'un programme en mémoire.

[Ctl\_Alt\_Z] tracer en mode T1.

[Sft\_Alt\_Help] arrêter le programme en cours d'exécution.

[Sft\_Ctl\_Alt\_Del]ete provoquer un reset à froid

[Sft\_Ctl\_Alt\_J]sr exécuter une routine en mode superutilisateur.

#### 3.3 Fenêtres

Les **fenêtres** d'Adebug sont au nombre de **5** (voir figure 1), numérotées de 1 à 5. La **fenêtre active** ou **fenêtre courante** est signalée par un cadre épais.

### [Alt Z]oom de fenêtre

Agrandir la fenêtre courante à la taille de l'écran. Celle-ci est alors la seule visible, mais toutes les fonctions restent disponibles. Un nouvel appui sur [Alt\_Z] ou sur [Escape] fait sortir du mode Zoom.

```
00121C18 OpenFLI
                           MOVEM.L D1-D2/A0-A2/A6,-(A7)
                        (>)MOVEA.L A1,A6
00121C1C
00121C1E
                           BSR FOpenFil
                           BMI.B $121C6A
SUBQ.W #4,A7
00121C22
                                                        Indicateur
00121C24
                                                        du PC
00121C26
                           MOVEA.L A7,A0 [1:-1]
                           MOVE.W D0,D1
MOVEQ #4,D0
00121C28
00121C2A
00121C2C
                           BSR FReadFil
                           MOVEM.L (A7)+,D2
BNE.B $121C72
ROR.W #8,D2
|00121C30
00121C36
00121C38
                           SWAP D2
                           ROR.W #8,D2
MOVE.L D2,D0
00121C3A
00121C3C
00121C3E
                           BSR FMalloc
                           BNE.B $121C72
MOVE.L A0,($30,A6)
MOVE.L D2,(A0)+
|00121C42
00121C44
00121C48
00121C4A
                           MOVE.L D2,D0
                           SUBQ.L #4,D0
00121C4C
00121C4E
                           BSR FReadfil
00121C52
                           BNE.B $121C72
```

Breakpoint #1 set.

Figure 2: une fenêtre en mode désassemblage plein écran ([Alt\_Z]oom de fenêtre).

#### [Tab]

Changer de fenêtre courante. L'ordre de sélection est : 1, 2, 3, 4, 5.

### [Alt\_S]épare/Regroupe fenêtres

Séparer une fenêtre en deux ou regrouper deux fenêtres en une. Sur les fenêtres 2 et 3:

Coupe la fenêtre en deux fenêtres (2 et 4, ou 3 et 5)

Sur les fenêtres 4 et 5 (ou 2 et 3):

Prend les deux fenêtres 2 et 4 (ou 3 et 5) et les réunit.

#### Sur la fenêtre 1:

La fait disparaître de l'écran. Pour la faire réapparaître, appuyer sur [Alt\_1].

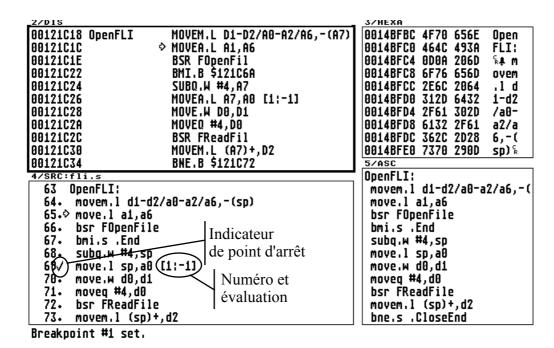


Figure 3: le mode quatre fenêtres par élimination de la fenêtre 1.

# [Alt\_T]ype de visualisation

Changer le type de visualisation de la fenêtre courante.

Sept types de visualisation sont possibles à l'intérieur d'une fenêtre:

Registres du 68xxx.

Désassemblage.

"Dump" hexadécimal.

"Dump" ASCII.

Source.

Variables.

Inspect.

### [0~7] Changer de type la fenêtre courante

0: type registres

1: type désassemblage

2: type hexadécimal

3: type ASCII

(4: non implémenté)

5: type source

6: type variables

7: type inspect

### [Alt 1~5]

Sélectionner la fenêtre 1~5. Équivalent donc à un certain nombre de [Tab], mais plus rapide pour passer de la fenêtre 2 à la fenêtre 1, par exemple...

► Si la fenêtre demandée n'est pas ouverte, elle apparaît alors automatiquement. C'est d'ailleurs le seul moyen de faire réapparaître la fenêtre 1 en cas de [Alt\_S] sur elle.

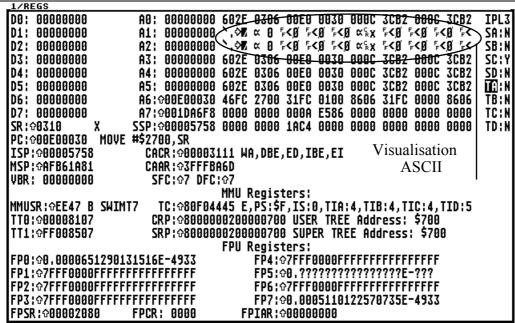
### [Ctl\_0~7]

Sur une fenêtre de type registres:

changer l'affichage du contenu de l'adresse pointée par le registre **A0~A6**) entre **hexadécimal** et **ASCII**.

Dans tous les autres cas:

positionner la fenêtre courante sur le registre A0~A7.



Current path changed.

Figure 4: le mode visualisation ASCII du contenu des registres.

### [Ctl 8]

Positionner la fenêtre courante sur le registre **SSP** (pile superviseur).

### [Ctl\_9]

Positionner la fenêtre courante sur le registre **PC** ("Program Counter" ou compteur ordinal).

### [Ctl Alt 1~5]

Positionner la fenêtre courante à l'adresse de la fenêtre 1~5.

#### [Sft\_Ctl\_0~7]

En mode désassemblage, mémoire et ASCII, positionne la fenêtre courante sur le contenu du registre **AO~A7**.

Si le registre contient une adresse illisible ou impaire, le message **Adresse impaire ou illisible.** apparaît.

En mode registres, passe l'affichage des registres **FPO~FP7** d'hexadécimal en flottant et vice-versa (sur une machine avec FPU).

### [Sft Ctl 8]

Positionner la fenêtre courante sur le contenu du registre **SSP** (pile superviseur).

Si SSP contient une adresse illisible ou impaire, le message **Adresse impaire ou illisible.** apparaît.

### [Sft Ctl 9]

Positionner la fenêtre courante sur le contenu du registre **PC** (compteur ordinal).

Si le PC contient une adresse illisible ou impaire, le message **Adresse impaire ou illisible.** apparaît.

### [Flèche haut]

Remonter l'adresse d'affichage de la fenêtre courante d'une ligne.

### [Flèche bas]

Redescendre l'adresse d'affichage de la fenêtre courante d'une ligne.

### [Flèche gauche]

En mode désassemblage, mémoire et ASCII, remonte l'adresse de base de la fenêtre courante de deux octets en type désassemblage et d'un octet dans le cas contraire.

En mode registres et source, décale l'affichage d'une colonne vers la gauche.

#### [Flèche droite]

En mode désassemblage, mémoire et ASCII, redescend l'adresse d'affichage de la fenêtre courante de deux octets en type désassemblage et d'un octet dans le cas contraire.

En mode registres et source, décale l'affichage d'une colonne vers la droite.

# [Ctl\_Flèche gauche]

Diminuer la taille de la fenêtre courante vers la gauche.

### [Ctl Flèche droite]

Agrandir la taille de la fenêtre courante vers la droite.

### [Sft\_Flèche haut]

Remonter l'adresse d'affichage de la fenêtre courante d'une page.

### [Sft\_Flèche bas]

Redescendre l'adresse d'affichage de la fenêtre courante d'une page.

### [Ctl\_Flèche gauche]

Agrandit vers la gauche la fenêtre courante.

### [Ctl\_Flèche droite]

Agrandit vers la droite la fenêtre courante.

### [Ctl\_Flèche haut]

Augmenter la hauteur de la fenêtre courante.

### [Ctl Flèche bas]

Diminuer la hauteur de la fenêtre courante.

### [A]dresse de fenêtre

Modifier l'adresse de base de la fenêtre courante.

#### 

met la fenêtre courante à l'adresse 3.

#### [Alt L]

Lier l'adresse de base de la fenêtre courante à une expression donnée.

#### [Alt\_L] Lie expression <e>: {4ba}

Lier la fenêtre courante avec le contenu de l'adresse \$4ba.

#### [W]atch

Si le PC pointe sur une instruction de changement de flux (Bcc,

Jcc, RTD, RTE, RTS, RTR, ILLEGAL, TRAP, etc...), permet de visualiser dans la fenêtre courante (de type désassemblage ou source) l'adresse de destination. Cette évaluation accepte tous les modes d'adressage autorisés par le 68xxx.

Sur une fenêtre de type variables et devant une variable de type agrégat (tableau, structure, union), passe en mode inspect sur celle-ci.

© Cette fonction est une bascule (toggle), un deuxième appui ramène à l'ancienne adresse.

✓ BSR \$123456

JSR 24(a3,a2.l)

### [Sft\_Watch]

Si le PC pointe sur une instruction de changement de flux (Bcc, Jcc, RTD, RTE, RTS, RTR, ILLEGAL, TRAP, etc...), permet de visualiser dans la fenêtre courante (de type désassemblage ou source) l'adresse de destination. Cette évaluation accepte tous les modes d'adressage autorisés par le 68xxx.

Sur une fenêtre de type variables et devant une variable de type agrégat (tableau, structure, union), passe en mode inspect sur celle-ci.

BSR.L \$20000 JSR ([24.W,a3,a2.1])

#### [#]

Sur une fenêtre en type source, permet d'aller au numéro de ligne donné.

✓ Goto line number <#>: 125

### [Ctl\_Home]

Sur une fenêtre en mode source, amène en début de fichier source.

Sur une fenêtre en mode variables, amène au début des variables.

### [Sft\_Ctl\_Home]

Sur une fenêtre en mode source, amène en fin de fichier source. Sur une fenêtre en mode variables, amène à la fin des variables.

### [Ctl\_Alt\_G]

Sur une fenêtre en mode variables, demande l'affichage dans celle-ci des variables globales.

### [Ctl\_Alt\_S]

Sur une fenêtre en mode variables, demande l'affichage dans celle-ci des variables statiques.

### [Ctl\_Alt\_L]

Sur une fenêtre en mode variables, demande l'affichage dans celle-ci des variables locales.

### [Ctl\_I]

Sur une fenêtre en mode inspect, demande l'affichage dans celleci du contenu de la variable (non pointeur) spécifiée.

Set inspect: my\_var

## 3.4 Gestion et édition mémoire

# [Alt\_E]dition de fenêtre

Sur une fenêtre en type **Registres du 68xxx**:

Passer en mode édition du **SR** et des 2 **timers A**, **B**, **C**, **D**. En mode édition du SR, les touches **T**, **S**, **X**, **N**, **Z**, **V**, **C** changent l'état du bit correspondant.

La touche **[Tab]** passe du SR aux timers, et vice-versa. Ceux-ci sont autorisés ou au contraire inhibés par les touches **[Y]**es et **[N]**o.

```
1/REGS
DO: 00000000
                      AO: 00000000 602E 0306 00E0 0030 000C 3CB2 000C 3CB2
                      D1: 00000000
                                                                                  SA:N
D2: 00000000
D3: 00000000
                                                                                 SB:N T1 B
SC:Y T1 C
                      A3: 00000000 602E 0306 00E0 0030 000C 3CB2 000C 3CB2
A4: 0000000 602E 0306 00E0 0030 000C 3CB2 000C 3CB2
D4: 00000000
D5: 00000000
                      A5: 00000000 602E 0306 00E0 0030 000C 3CB2 000C 3CB2
                                                                                  TB: N T2 B
D6: 00000000
D7: 00000000
                      A6:000E00030 46FC 2700 31FC 0100 8606 31FC 0000 8606
A7:0001DA6F8 0000 0000 000A E586 0000 0000 0000 0000
                                                                                 TC:N
SR:00310
                     SSP:000005758 0000 0000 1AC4 0000 0000 0000 0000 0000
                                                                                 TD:NT2 D
PC:000E00030 MOVE #$2700,SR
ISP:☆00005758
                       CACR:000003111 WA,DBE,ED,IBE,EI
                                                                         Curseur
MSP:☆AFB61A81
                       CAAR: ☆3FFFBA6D
                                                                         d'édition
VBR: 00000000
                        SFC:07 DFC:07
MMU Registers:
MMUSR:⊕EE47 B SWIMT7 TC:⊕80F04445 E,PS:$F,IS:0,TIA:4,TIB:4,TIC:4,TID:5
                        CRP:08000000200000700 USER TREE Address: $700
TT0:000008107
                        SRP:08000000200000700 SUPER TREE Address: $700
TT1:0FF008507
                                    FPU Registers:
FP0:00.0000651290131516E-4933
                                          FP4:☆7FFF0000FFFFFFFFFFFFFF
FP1:☆7FFF0000FFFFFFFFFFFFFFF
                                          FP5:00.777777777777777
FP2:07FFF0000FFFFFFFFFFFFFFF
                                          FP6:07FFF0000FFFFFFFFFFFFFFF
FP3:07FFF0000FFFFFFFFFFFFFFF
                                          FP7:00.0005110122570735E-4933
FPSR:000002080
                     FPCR: 0000
                                        FPIAR: 000000000
Current path changed.
```

Figure 5: le mode édition du SR et des timers ([Alt\_E]).

Sur une fenêtre en type désassemblage:

Permet d'assembler "en ligne" toute instruction 68000.

La base numérique par défaut est le décimal, bien sûr. Pour un MOVEM, il faut taper movem.l d0-d7/a0-a6,... et non pas movem.l d0-a6,...

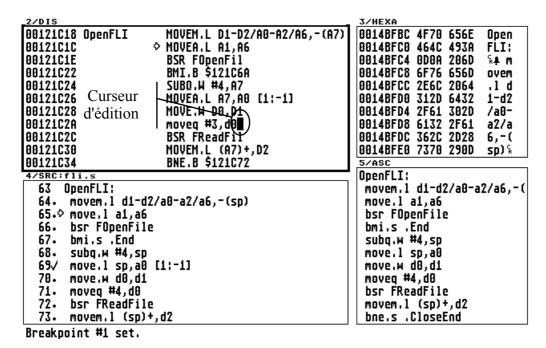


Figure 6: le mode assemblage en ligne ([Alt\_E]).

Sur une fenêtre en type 'dump' hexadécimal:

Passer en mode édition de la mémoire. Les touches **0-9** ainsi que **A-F** modifient le contenu de la mémoire.

La touche **[Tab]** passe du champ hexadécimal au champ ASCII. Le champ ASCII est éditable avec toutes les touches. Le curseur peut être déplacé grâce aux flèches haut, bas, gauche et droite.

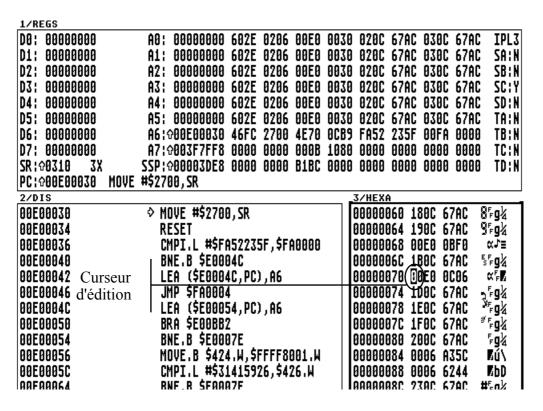


Figure 7: le mode édition de mémoire en hexadécimal ([Alt\_E]).

Sur une fenêtre de type '**ASCII**' : Permet d'éditer la mémoire en ASCII.

L'édition mémoire se fait en mode remplacement et non insert.

```
1/REGS
DO: 00000000
                  A0: 00000000 602E 0206 00E0 0030 020C 67AC 030C 67AC
D1: 00000000
                  A1: 00000000 602E 0206 00E0 0030 020C 67AC 030C 67AC
                                                                     SA:N
D2: 00000000
                  A2: 00000000 602E 0206 00E0 0030 020C 67AC 030C 67AC
                                                                     SB:N
D3: 00000000
                  A3: 00000000 602E 0206 00E0 0030 020C 67AC 030C 67AC
D4: 00000000
                  A4: 00000000 602E 0206 00E0 0030 020C 67AC 030C 67AC
                                                                     SD:N
                  A5: 00000000 602E 0206 00E0 0030 020C 67AC 030C 67AC
D5: 00000000
D6: 00000000
                  A6:000E00030 46FC 2700 4E70 0CB9 FA52 235F 00FA 0000
                                                                     TB:N
D7: 00000000
                  TC:N
SR:00310 3X
                  TD:N
PC:000E00030 MOVE #$2700,SR
 IFNE MC68000
 move.b 3(a0),d0
 lsl.w #8,d0
 move.b 2(a0),d0
 swap d0
move.b 1(<del>a</del>Q),d0
lsl.w #8Md0
move.b (a0),d0
                         Curseur
                        d'édition
 ELSEIF
 move.l (a0),d0
 ror.w #8,d0
 swap d0
```

File <FLI.S> loaded in 124242 bytes.

Figure 8: le mode édition de mémoire en ASCII ([Alt\_E]).

# [C]opier

Copier un bloc de mémoire (source, longueur, destination). Si un ou plusieurs octets de la destination ne sont pas inscriptibles, le message d'avertissement

# Attention! Copie incomplète.

s'affiche. Néanmoins, les autres octets auront été écrits.

Il est possible de copier une zone sur une partie d'elle-même

# [F]ill

Remplir un bloc de mémoire (destination, longueur, valeur). Si un ou plusieurs octets ne sont pas inscriptibles, le message d'avertissement :

## Attention! Remplissage incomplet.

s'affiche. Néanmoins, les autres octets auront été remplis. La valeur de remplissage est traitée en octet.

F [F] Remplir <destination>,<longueur>,<valeur>: 8,ff,0 remplit \$ff octets à partir de l'adresse \$8 avec 0.

Vous pouvez aussi remplir un bloc avec une chaîne ASCII en l'encadrant avec des guillemets (" ").

F [F] Remplir <destination>, <longueur>, <valeur>: 8,ff,"tototata" remplit \$ff octets à partir de l'adresse \$8 avec la chaîne "tototata".

# [G]et expression

Rechercher en mémoire un octet, mot, long mot, chaîne ASCII ou instruction.

La recherche commence à partir de l'adresse de la fenêtre courante.

La touche **[Esc]** force l'arrêt de la recherche, même si l'expression recherchée n'a pas été trouvée.

Quand la recherche est réussie, la fenêtre courante est rafraîchie à l'adresse trouvée.

Même les mots ou longs mots commençant à une adresse impaire seront trouvés.

De plus, il est possible de rechercher dans la mémoire non lisible.

Pour la recherche d'instruction, vérifiez que vous écrivez les noms des instructions en majuscules et sans tabulations. En effet, la distinction entre majuscule et minuscule est faite.

## **☞** [G] L)ong Cherche: 12345678

cherche le long mot de valeur \$12345678 à partir de l'adresse de la fenêtre courante.

# [N]ext

Rechercher l'occurrence suivante de l'expression donnée dans **[G]**. **[Esc]** force l'arrêt de la recherche.

# [Alt N]

Rechercher l'occurrence précédente de l'expression donnée dans [G].

[Esc] force l'arrêt de la recherche.

## 3.5 Macros

Le but d'une **macro** est d'éviter de refaire à la main plusieurs fois la même manipulation. Il ne s'agit pas d'un réel langage de commande, même si certaines fonctionnalités s'en approchent. Une macro est avant tout une représentation symbolique des entrées clavier d'Adebug. Chaque touche associée à une fonction est représentée par

Le signe ` (backquote).

L'état des touches: sft\_, alt\_, ctl\_.

La lettre correspondant à la touche du clavier.

`ctl\_d `ctl\_j `sft\_up `right...

La macro est stockée en ASCII. Il est donc possible de créer ou de modifier une macro sous un éditeur. Mais il est souvent plus simple d'enregistrer la séquence de manipulation grâce à la fonction [M]acro **E)nregistrement**.

La structure du fichier macro est assez libre:

Une ligne peut contenir un nombre indéfini de macros, à concurrence de 200 caractères par ligne. Une macro ne peut être à cheval sur plusieurs lignes.

`ctl\_b est valide.

`ct

l\_b n'est pas valide.

On peut introduire un commentaire en faisant commencer la ligne par ;.

; ceci est un commentaire

Dans le cas d'une fonction demandant une expression dans la ligne de commande, on peut utiliser un point d'interrogation (?) pour rentrer une expression à la main.

✓ `alt\_b ? provoquera l'entrée dans la fonction [Alt\_B] et l'attente d'une expression dans la ligne de commande.

# [M]acro

Donne accès aux menus de contrôle des macros.

## E)nregistrer J)ouer C)harger V)oir une macro.

E)nregistrer donne accès au sous menu suivant:

## E)nregistrer T)racer A)rrière P)asser C)lr

# **Enregistrer**

Lancer l'enregistrement de macro.

## Tracer

Lancer l'enregistrement d'une commande.

## Arrière

Revenir en arrière d'une commande.

## Passer

Exécuter une commande sans l'enregistrer.

## Clr

Effacer le tampon de macro.

J)ouer fait accéder au sous menu suivant:

# J)ouer T)racer A)rrière P)asser V)oir D)ébut

# Jouer

Lancer l'exécution de la macro en mémoire.

# Tracer

Lancer l'exécution de la prochaine commande de la macro.

# Arrière

Revenir en arrière d'une commande.

## Passer

Sauter la prochaine commande.

# Voir

Visualiser la macro à partir de la prochaine commande.

## Début

Revenir au début du tampon de macro.

## C)harger

Demande un nom de fichier de macro et le charge.

## V)oir

Visualiser la totalité du tampon de macro.

On peut contrôler son affichage à l'aide des touches suivantes:

[Flèche bas] ou [>]: descendre d'une ligne. [Flèche haut] ou [<]: remonter d'une ligne. descendre d'une page. [Sft\_Flèche bas] ou [Espace]: [Sft\_Flèche haut] ou [Sft\_Espace]: remonter d'une page. [Homel: amène en début de macro. [Sft\_Home]: amène en fin de macro. [P]: impression de la macro. [S]: sauvegarde de la macro.

[M] fait apparaître le menu des macros.

E)nregistrement passe dans le menu d'enregistrement.

E)nregistrement passe en mode d'enregistrement.

Dorénavant, chacune de vos actions dans Adebug est enregistrée dans le tampon **MAC**. Par exemple, tapez [V] plusieurs fois. Pour arrêter l'enregistrement, il suffit de repasser dans le menu macro ([M]).

L'option **V)oir** du menu macro permet de voir la macro en entier. Dans ce mode, la touche [S] donne accès à la sauvegarde sur disque.

Pour jouer la macro, il faut entrer dans le menu correspondant avec l'option **J)ouer**.

Cette dernière permet de rejouer la macro à partir de la position courante.

Vous devez voir l'écran clignoter car une macro jouée va beaucoup plus vite que son équivalent réalisé au clavier.

Si vous avez précédemment sauvé la macro sur disque sous un nom quelconque, vous pouvez la recharger avec l'option **C)harger** du menu de macro.

Vous pouvez aussi reprendre le fichier directement en ASCII avec un éditeur de texte et le modifier en respectant la syntaxe des macros.

De cette manière, il est possible de rajouter des fonctions de contrôle dans une macro. Chacune de ces fonctions débute par un °, et consiste en une lettre:

°c <texte> permet d'introduire un commentaire (sans espace) qui sera affiché avec la possibilité d'interrompre l'exécution de la macro.

°c Arrêter\_la\_macro\_?

donnera

## Arrêter\_la\_macro\_?. Continue Y)es/N)o?

**°F** entraîne, en cas de message d'erreur, une pause dans l'exécution de la macro avec possibilité d'interrompre son exécution.

## Fichier non trouvé. Continue Y)es/N)o?

°f force à continuer l'exécution de la macro en cas de message d'erreur.

(Par défaut, le mode °F est sélectionné.)

**°t** provoque le passage en mode trace de la macro.

°T relance l'exécution en mode normal.

°s arrête directement l'exécution.

°1 <étiquette> sert à déclarer une étiquette (label).

**°b <expression> <étiquette>** continue l'exécution à partir de <étiquette> si <expression> est vraie (-1), sinon exécute la commande suivante.

Trace jusqu'à ce que d0 soit égal à 0.

## 3.6 Variables

# [Alt\_V]ariable

Créer une variable de n'importe quel type, en utilisant la même syntaxe que dans le fichier adebug.var:

Nom de la variable: Une suite d'au maximum 79 caractères ASCII quelconques hormis les symboles réservés pour l'évaluateur:

(	)	[	]							
!	~	++		+	_	*	&	sizeof		
*	/	%								
+	ı									
<<	<b>^</b>									
<	<=	>	>=							
==	!=									
&										
^										
&&										
5:										
=	+=	-=	*=	/=	%=	&=	^=	=	<<=	>>=
,										

Une variable peut être de six types:

## LA: Label absolu

Ce type permet de référencer des variables numériques simples comme des adresses ou d'autres valeurs constantes. Une fois initialisée, une LA possède toujours la même valeur (elle n'est plus jamais réévaluée).

La VAR reset\_pc est de type LA, et vaut le contenu de \$4.

# **EQ**: Label équivalent

Ce type permet de référencer des variables numériques simples comme des offsets de structures ou d'autres valeurs constantes. Une fois initialisée, un EQ possède toujours la même valeur (il n'est plus jamais réévalué).

# struct\_next,eq=8

La VAR struct\_next est de type EQ, et sa valeur est 8.

### LR: Label relatif

A la différence des LA, les LR sont réévalués à chaque demande, et la nouvelle valeur est renvoyée.

## count\_200,lr={4ba}

La VAR count\_200 est de type LR et sa valeur est réévaluée à chaque demande:

{4ba} donnera le contenu du compteur 200Hz du système.

#### BL: Bloc

Tout fichier chargé en mémoire par Adebug (sauf adebug.sav, adebug.var, les fichiers de macro et le programme en cours de débogage) se trouve référencé sous forme de VAR BL. Avec le nom du bloc (qui est le nom du fichier sur disque) se trouvent mémorisées les adresses de début et de fin en mémoire du fichier chargé.

Sous l'évaluateur, le nom du bloc renvoie son adresse de début. Pour connaître l'adresse de fin, il faut consulter la liste des VAR (fonction [L]).

## newblock.bl='data.bin

Initialise le BL newblock par le chargement du fichier data.bin

#### **RO**: Routine

Une RO est avant tout un bloc, c. à. d. qu'elle se trouve mémorisée comme tel. En revanche, elle sera exécutée à l'évaluation de son nom. Une RO est donc un fichier binaire exécutable, relogé au chargement et dont la section BSS est allouée. Elle est appelée en mode utilisateur, avec des piles utilisateur et superviseur qui lui sont propres.

## 

Initialise la RO linea par le chargement du fichier linea.ro Afin de retourner dans Adebug, elle doit se terminer par un RTS (la pile ne doit donc pas être décalée) et non par **Pterm**. Cette routine est libre d'effectuer n'importe quelle opération à l'intérieur d'elle même.

Elle peut néanmoins communiquer avec Adebug. Elle reçoit à l'exécution dans le registre **AO** l'adresse d'une structure de

communication (le source suivant est fourni sur la disquette).

ROSTRUCT.S Structure de communication entre une routine (RO) et Adebug.

RO:	rsstru	uct	
adbg_v	rs.w	1	;numéro de version d'Adebug
sys_vrs.w	1	;num	éro de version du système
w1	rs.l	1	;adresse de la fenêtre 1
w2	rs.l	1	;adresse de la fenêtre 2
w3	rs.l	1	;adresse de la fenêtre 3
w4	rs.l	1	;adresse de la fenêtre 4
w5	rs.l	1	;adresse de la fenêtre 5
text	rs.l	1	;adresse de la section TEXT du programme
	_		;en cours de débogage
data	rs.l	1	;adresse de la section DATA du programme
5.5.15.		-	;en cours de débogage
bss	rs.l	1	;adresse de la section BSS du programme
		•	;en cours de débogage
end	rs.l	1	;adresse de fin du programme
Ond	10.1	•	;en cours de débogage
valeur des	reais	tres d	u programme en cours de débogage
d0_reg	rs.l	1	a programme on coare ac accogage
d1_reg		-	
d2_reg	rs.l	_	
d2_reg	rs.l	1	
d3_reg d4_reg	rs.l		
	rs.l	1	
d5_reg		1	
d6_reg		1	
d7_reg	rs.l	1	
a0_reg	rs.l	_	
a1_reg	rs.l		
a2_reg	rs.l	1	
a3_reg	rs.l	1	
a4_reg	rs.l	1	
a5_reg	rs.l	1	
a6_reg	rs.l	1	
a7_reg	rs.l	1	
ssp_reg	rs.l	1	
sr_reg	rs.w	1	
pc_reg	rs.l	1	
reso	rs.w	1	résolution du programme en cours de
hasenage	re l	1	;débogage
basepage	rs.l	ı	;adresse de la basepage du programme en
ro addr	re l	1	;cours de débogage
ro_addr	rs.l	1	;adresse de la routine en cours d'exécution
screen	rs.l	1	;adresse de l'écran du programme en cours

			;de débogage
string rs.l	1	;cont	tient l'adresse d'une chaîne à afficher au
			;retour de la RO (ou sinon 0)
reserved	rs.l	2	réservé pour extensions futures;
reput_exc	rs.b	1	;drapeau signalant à Adebug de réinstaller
			;toutes les exceptions
IPL7	rs.b	1	;drapeau signalant à Adebug de passer en
			;IPL7
timer1a	rs.b	1	;drapeau marche ou arrêt du timer 1 A
timer1b	rs.b	1	;drapeau marche ou arrêt du timer 1 B
timer1c	rs.b	1	;drapeau marche ou arrêt du timer 1 C
timer1d	rs.b	1	;drapeau marche ou arrêt du timer 1 D
timer2a	rs.b	1	;drapeau marche ou arrêt du timer 2 A
timer2b	rs.b	1	;drapeau marche ou arrêt du timer 2 B
timer2c	rs.b	1	;drapeau marche ou arrêt du timer 2 C
timer2d	rs.b	1	;drapeau marche ou arrêt du timer 2 D
redraw	rs.b	1	;drapeau signalant à Adebug de redessiner
			;son écran
serial rs.b	1	;drap	peau signalant à Adebug de passer en
		-	;sortie série
sizeof rssizeof			
rsend			

De plus, la routine doit pouvoir indiquer quelque chose à l'utilisateur. Ainsi, la valeur renvoyée dans le registre **D0** en sortie de la **R0** sera prise en compte comme résultat de l'évaluation.

L'utilisateur peut aussi communiquer avec la **RO** en lui passant des paramètres manuellement. Dans ce cas, c'est l'évaluation de chacun des paramètres qui est passée à la routine sous forme d'un tableau de mots longs dont l'adresse se trouve dans le registre **A1** et le nombre d'éléments dans le registre **D0**.

voir le source de la routine ligne.s fourni sur la disquette.

### **EX**: Variable existant:

Les **EX** sont des LA relatives au début de la section à laquelle elles appartiennent. Elles permettent de créer des symboles dont les valeurs sont indépendantes des adresses de chargement des différentes sections du programme débogué. Ainsi, si l'on crée et l'on sauve des variables de type **EX** lors d'une session de travail avec Adebug, on pourra plus tard en rechargeant ces variables continuer à travailler sur le même programme avec des labels corrects, même si l'adresse de chargement du programme est différente.

## 

Initialise l'EX basepage à la valeur TEXT-\$100.

# adebug.var:

Fichier ASCII contenant les définitions de VARs que l'on désire initialiser au chargement d'Adebug. Chaque ligne est indépendante (autrement dit, une déclaration de VAR ou un commentaire ne peuvent excéder une ligne de 80 caractères).

## On peut écrire:

- soit une déclaration de VAR:

## <nom de la var>,<type>=<expression>

soit une remarque commençant par un;

- soit une option:

Elles sont annoncées par le signe '-' (moins) en début de ligne. Elles sont autorisées ou interdites par les signes '+' (plus) et '-' (moins).

Trois options existent:

- **d** attend la pression d'une touche en cas d'erreur signalée à l'initialisation d'une variable.
- **D** attend la pression d'une touche à chaque initialisation de variable.
- interdit la vérification de l'existence de la variable avant sa création (à manipuler avec précaution!).
  - ✓ -0+ enlève la vérification de la préexistence des variables.
- soit une ligne vide.
  - [Esc] permet d'interrompre les initialisations.
- [Sft\_Sft] au lancement permet d'éviter le chargement par défaut du fichier ADEBUG.VAR.

# [Ctl\_Alt\_V] Charger les symboles

Sert à charger les symboles d'un programme en cours de route. Cette commande est surtout utile dans Rdebug pour déboguer une tâche qui n'a pas été chargée sous Adebug.

# [Ctl\_V]

Permet de charger un fichier de variables en cours de route.

# [Help]

Visualiser les informations générales.

Cette fonction comporte trois parties à l'affichage:

Première partie: Les informations internes, à savoir

- Les nombres maximum et courant des points d'arrêt, des blocs, et des instructions enregistrées dans l'historique (STO) ainsi que les tailles maximale et courante des tampons VAR, LA, LR, BL, EX, MAC, HIS.
- Les adresses de début et de fin de la zone de données interne d'Adebug.

Deuxième partie: Les informations sur le matériel, en l'occurence le microprocesseur présent et sa fréquence estimée.

Troisième partie: Le nom du programme en cours de débogage, ainsi que les adresses et les longueurs de ses sections.

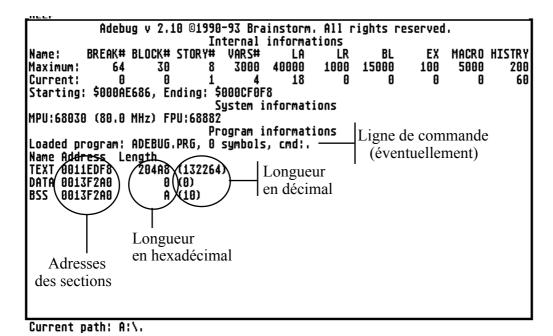


Figure 9: liste des tailles courante et maximale des tampons de variables.

# [L]iste des variables

Afficher toutes les variables.

Les variables sont affichées dans le format suivant:

## <nom>,<type>=<expression>

<expression> dépend du type:

LA: Leur valeur.

LR: L'expression à évaluer. RO: L'adresse de la routine.

BL: Les adresses de début et de fin du bloc ainsi que sa taille.

EX: Leur valeur.

Dans le cas où la variable est prise en compte par l'échantillonneur de temps d'exécution (voir le chapitre **Profiler**), deux informations supplémentaires figurent au début de la ligne: 

compteur
qui rendent compte du temps
machine utilisé dans la portion de code suivant la variable.
Les pages sont numérotées sur la première ligne de la fenêtre.

On peut contrôler l'affichage de la liste à l'aide des touches suivantes:

```
[Flèche bas] ou [>]:
                                      descendre d'une ligne.
[Flèche haut] ou [<]:
                                      remonter d'une ligne.
[Sft_Flèche bas] ou [Espace]:
                                      descendre d'une page.
[Sft_Flèche haut] ou [Sft_Espace]:
                                      remonter d'une page.
                                      amène en début de liste.
[Home]:
[Sft_Home]:
                                      amène en fin de liste.
[P]:
                                      impression de la liste.
[S]:
                                      sauvegarde de la liste.
```

```
VARIABLES
 ;page #4
  0.00%
                  0 FSelAnyT,la=$12A5B6
                  8 FSelTxt,la=$12A5AB
8 Fgetdta,la=$129F0E
 0.00%
                                                Pourcentage de temps CPU
  0.00%
  0.00%
                  O FileSele,la=$128EFE
                  0 FillBloc,la=$123564
0 FormAler,la=$129138
  0.00%
  0.00%
                 (B)FreeMous, la=$1292BC
  0.00%
                                                Nombre de "ticks"
  0.00%
                  Ø FreezeMu,la=$1280A2
                  0 FreezeWi,la=$128072
0 Fsetdta la=$129EDE
  0.00%
  0.00%
                                                Nom de la variable
                  0 Fsfirst, la=$129EEE
  0.00%
                  0 GEMExit, la=$128ADC
0 GEMInit, la<del>)$1289DE</del>
  0.00%
  0.00%
                                                Type de la variable
  0.00%
                  0 GetCurFs, 1a=$128E68
                  0 GetCurPa,la=$128E32
0 GetCurrC,la=$128258
0 GetCurrW,la=$128364
  0.00%
                                                Valeur de la variable
  0.00%
                                                   en hexadécimal
  0.00%
                  0 GetExtCo,la=$128318
  0.00%
  0.00%
                  0 GetFLIBr, la=$121D90
  0.00%
                  0 GetFLICo,la=$121D8A
  0.00%
                  0 GetIntCo,la=$1282CC
  0.00%
                  0 GetScree, la=$128226
```

File <B.PRG> loaded in 49972 bytes.

Figure 10: liste des variables ([L]).

# 3.7 Écran

# [V]oir

Inverser les **écrans** d'Adebug et du programme débogué. Toutes les fonctions d'Adebug restent disponibles quelque soit l'écran visualisé.

[Ctl\_Alt\_I]nverse Inverser les couleurs d'encre et de fond d'Adebug.

# 3.8 Disque

# [D]irectory

Changer de répertoire courant.

Syntaxe:

[<nom de lecteur>:\]<nom de répertoire>[\<nom de répertoire>]...

🗵 Le système d'exploitation ne supporte pas l'utilisation des caractères joker (\* et ?).

## [D] Répertoire: ros

Change le répertoire courant en ros.

# [Alt\_D]irectory

Afficher le **répertoire** courant et son contenu.

S'il y a plus de fichiers que n'en contient la fenêtre, appuyer sur une touche permet d'accéder à la suite du répertoire.

**[Esc]** interrompt.

[P] permet d'imprimer le contenu du répertoire courant.

# [Sft\_Alt\_D]irectory

Permet de fixer la sélection avant l'affichage du **répertoire**.

**=**>

[Sft\_Alt\_D] Sélection: \*.s

ne montrera que les fichiers dont l'extension est S.

# [Ctl\_L]oad program

Charger un fichier relogeable ou programme.

Cette fonction demande un nom de **programme exécutable**, éventuellement suivi d'une ligne de commande séparée de ce dernier par une virgule. Le nom peut comprendre des caractères joker (\* et ?) et ne pas contenir d'extension. Dans ce dernier cas, les extensions **prg**, **tos**, **app**, **ttp** et **gtp** lui seront successivement ajoutées. Le programme correspondant est chargé et relogé. Ensuite, les symboles de débogage sont chargés s'ils sont présents. Pour cela, il faut que le fichier exécutable contienne des symboles DRI (8 caractères) et/ou Extended Debug (22 caractères).

Si le programme chargé porte l'extension .TOS ou .TTP, l'écran logique est effacé, et le curseur allumé.

Deux fichiers sont éventuellement chargés après le programme.

Le fichier de sauvegarde des fenêtres et de l'historique:

<nomdeprogramme>.pri

La macro associée:

<nomdeprogramme>.mac

pour l'exécuter.

## [Ctl\_L] Charger prg: exemple

Charge le programme exemple.prg.

Adebug chargera exemple.pri s'il est présent, puis si exemple.mac existe, il la chargera et l'exécutera.

► Une fois qu'un programme a été chargé avec cette commande, il est impossible de la réutiliser avant la fin du programme à déboguer.

# [B]inary load

Charger un fichier binaire.

Cette fonction permet de charger un fichier binaire directement sans aucune modification mémoire (pas de relocation). Elle peut être utilisée de deux manières. La première consiste à ne fournir que le nom de fichier, Adebug lui allouant en mémoire la taille qu'il a sur disque et le chargeant à l'adresse fournie par l'allocation. L'autre méthode demande, en sus du nom de fichier, une adresse où le charger en mémoire. Il n'y a pas d'allocation mémoire dans ce cas.

Dans tous les cas, si le chargement s'est bien passé, l'adresse de la fenêtre courante est modifiée pour correspondre à l'adresse de chargement. Le fichier chargé est référencé comme variable **bloc**. On y accède par le nom du fichier.

## **☞** [B] Charger binaire: data.bin

Charge le fichier data.bin à l'adresse résultant de l'allocation mémoire.

## **☞** [B] Charger binaire: screen.bin,{44e}

Charge le fichier screen.bin à l'adresse {\$44e}.

# [Alt\_A]SCII

Charger un fichier **ASCII**.

Cette fonction est identique à la fonction précédente [B], mais une fois celle-ci terminée elle change le type de la fenêtre courante en

### visualisation ASCII.

[Alt\_A] Charger ASCII: adebug.var

Charge le fichier ASCII adebug.var.

# [S]ave binary

Sauver un fichier binaire.

Le but de cette fonction est de sauver sur disque une partie de la mémoire. Aussi, demande-t-elle un nom de fichier, sous lequel sera sauvé le bloc mémoire, une adresse de début de sauvegarde et la longueur à sauver en octets.

## 

Sauve les 32000 octets de l'écran logique sur disque sous le nom screen.bin.

# 3.9 Configuration

A presque chacun des types de **VAR** expliqués dans le chapitre sur les variables, correspond un **tampon** de stockage et donc une taille de tampon prédéfinie. C'est à l'utilisateur de déterminer la taille des divers tampons en fonction de ses besoins.

Le tampon **VAR** est commun à toutes les variables. Il contient leurs références, chacune occupant 10 octets.

Le tampon **LA** contient le nom de toutes les variables (de 1 à 79 caractères).

Le tampon **LR** contient les expressions à évaluer de toutes les LR (de 1 à 79 caractères).

Le tampon **BL** contient tous les BL (y compris les RO). Chaque BL occupe 10 octets en plus de sa propre taille.

Il existe une deuxième sorte de tampon, non lié au variables.

Le tampon **HIS** contient l'historique de la ligne de commande. Une commande enregistrée occupe au moins 1 octet.

Le nombre **STO** est le nombre d'éléments réservés pour l'historique des instructions. Chaque instruction mémorisée occupe 118 octets.

Le tampon **MAC** contient la macro en mémoire. Il est impossible de charger ou d'enregistrer une macro de taille supérieure à celuici.

Toutes les tailles de tampons précédemment mentionnées, ainsi que diverses informations, sont sauvegardées, si l'utilisateur le souhaite, dans le fichier **adebug.sav**, lu au lancement d'Adebug.

# [Ctl\_P]références

Modification et sauvegarde des préférences d'Adebug.

Liste des préférences, et de leur valeur par défaut:

Trace pour entrer dans les exceptions qui le permettent: (Trap, Division par zéro, Check)

Défaut: Non

Réinstalle l'adresse de l'exception Trace avant de tracer:

Défaut: Oui

Commute l'écran d'Adebug et l'écran logique en cas de [Ctl\_A]:

Défaut: Oui

Commute l'écran d'Adebug et l'écran logique en cas de [Ctl\_R]:

Défaut: Oui

Commute l'écran d'Adebug et l'écran logique en cas d'exception Line A-F, Division par zéro, Check, Trapv:

Défaut: Non

Affiche les symboles (s'il y en a) à la place des valeurs dans les instructions comportant une partie en relatif registre:

Défaut: Oui

Affiche les symboles en mode désassemblage:

Défaut: Oui

Noms de variable différencié majuscule/minuscule

L'évaluateur ne fait pas de distinction majuscule/minuscule pour les noms de variables.

Défaut: Oui

Symboles de débogage uniquement

Les seuls symboles affichés sont les symboles du programme en cours de débogage.

Défaut: Non

Symboles DRI absolu

Les symboles DRI (ou extended) marqués DATA et BSS seront considérés comme TEXT (utile avec les linkers PLINK et ALN).

Défaut: Non

### Vérifier la date/heure des modules

La date/heure de chaque module sera comparée à la date/heure de l'exécutable, et un avertissement sera généré si la première est postérieure à la seconde.

Défaut: Oui

## Afficher les marques de correspondance code/source

Un symbole • est affiché à chaque ligne de source correspondant effectivement à du code.

Défaut: Oui

## Run until main au chargement

Exécuter le programme jusqu'au symbole main au chargement.

Défaut: Oui

## Césure de ligne automatique

Si positionné, l'affichage ASCII passe à la ligne suivante si elle est plus longue que la largeur de la fenêtre.

Défaut: Non

## Montrer les caractères invisibles

Permet de voir les tabs, espaces, retour chariot et saut de ligne.

Défaut: Non

## Sauver les informations du programme

Entraîne la sauvegarde des fenêtres et de l'historique au moment de quitter le programme en cours de débogage, sous le nom <nomduprogramme>.pri

Ce fichier sera rechargé au prochain chargement du programme.

Défaut: Non

#### Inverse vidéo

Si positionné, l'affichage d'Adebug se fera en inverse vidéo.

Défaut: Oui

## Autorise le temps réel:

Défaut: Oui

## N'autorise le temps réel que dans la fenêtre courante:

Défaut: Non

## Autorise le lien des fenêtres en temps réel:

Défaut: Non

# Signale si la sortie série doit être effectuée en émulation Minitel:

Défaut: Oui

## Commute l'affichage d'Adebug sur port série à son lancement:

Défaut: Non

## Pas d'opération sur disque dur en IPL > 5:

Les accès disque dur nécessitent le fonctionnement des interruptions. Ceux-ci ne seront pas autorisés si IPL interne d'Adebug est supérieur à 5.

Défaut: Oui

## Vérification des vecteurs disque systèmes

Vérifie si les vecteurs disque sont corrects avant d'exécuter tout accès disque.

Défaut: No

## Impression système

Toute sortie sur imprimante se fera à travers les fonctions systèmes standards (Adebug n'imprimant que sur port parallèle, ceci permet donc l'impression sur un autre port).

Défaut: Oui

### Suivi de l'IPL

L'IPL interne d'Adebug suit automatiquement celui du programme en cours de débogage. Ceci permet de planter dans une interruption, par exemple, et de toujours avoir Adebug au même niveau d'interruption.

Défaut: Oui

## Détourner division par zéro

L'exception Division par Zéro sera détournée.

Défaut: No

## Détourner ligne F

L'exception Ligne F sera détournée (utile surtout autre que 68000).

Défaut: No

## Détourner violation de privilège

L'exception Violation de Privilège sera détournée.

Défaut: Yes

## Échantillonneur de temps d'exécution actif

Les compteurs associés à chaque variable sont incrémentés en fonction de la position du PC à chaque interruption Vbl.

Défaut: Oui

Échantillonneur de temps d'exécution pour tous les symboles Si oui, toutes les variables sont échantillonnées, sinon uniquement celles appartenant au programme.

Défaut: Non

# Valeur de tabulation pour l'affichage de type ASCII dans les fenêtres:

Défaut: 8

## Longueur d'une ligne ASCII pour l'affichage dans les fenêtres:

Défaut: \255

## Numéro de vecteur par défaut pour les points d'arrêt:

Défaut: \32, vecteur correspondant à TRAP #0

## IPL au lancement d'Adebug:

Défaut: 3

# Vitesse du port série: (voir l'explication dans l'appendice sur le port série)

Défaut: 2. (Soit 4800 bauds, pour le Minitel 1B)

# Parité du port série: (voir l'explication dans l'appendice sur le port série)

Défaut: \$be

# Nombre maximum de points d'arrêt:

Défaut: \64

## Nombre maximum de blocs:

Défaut: \30

# Nombre d'entrées dans l'historique des instructions (STO):

Défaut: 8

# Nombre maximum de variables (VAR):

Défaut: \3000

# Taille du tampon de noms de variables (LA):

Défaut: \10000

# Taille du tampon de variables relatives (LR):

Défaut: \1000

## Taille du tampon de blocs (BL):

Défaut: \15000

## Taille du tampon de variables existant (EX):

Défaut: \100

## Taille du tampon de macro (MAC):

Défaut: \5000

# Taille du tampon de l'historique de la ligne de commande (HIS):

Défaut: \200

# 3.10 Opérations de contrôle de flux du programme

# [Ctl\_C]

Sort du programme en cours de débogage, ou sort d'Adebug si aucun programme n'est en cours de débogage. En sortant, Adebug réinstalle les vecteurs du système (de \$8 à \$140), ainsi que l'écran, la résolution, la palette, etc...

- Néanmoins, si le programme en cours de débogage a modifié certaines variables internes au système de l'Atari, l'ordinateur peut 'planter' et afficher des bombes.
- 🗵 Interrompre un programme GEM dans une boucle de multitâche ( dans Evnt\_Multi par exemple ) provoque généralement un 'plantage' lors du prochain recours au multitâche.

# [Sft Ctl C]

Comme ci-dessus, mais exécute l'appel wind\_new() juste avant. Cet appel AES permet la réinitialisation des fenêtres et d'autres choses...

- wind\_new() ne marche qu'à partir du TOS 1.4.
- 🖄 Exécuter cette fonction en étant déjà dans un appel AES va "halter" le système.

# [Ctl Z]

**Tracer** une seule instruction.

Une fois cette dernière exécutée, les fenêtres sont redessinées. La fenêtre 2 contient en permanence l'instruction pointée par le **PC**.

# [Ctl S]kip

Passer à l'instruction suivante sans exécuter l'instruction courante.

# [Ctl\_R]un

Lancer le programme à partir du PC.

Un **drapeau** dans les préférences permet de visualiser l'écran logique durant son exécution.

# [Ctl A]rrêt

Place un point d'arrêt à l'instruction suivante et lance le programme (particulièrement utile pour les instructions de saut avec retour BSR et JSR).

Un **drapeau** dans les préférences permet de visualiser l'écran logique durant son exécution.

# [T]race (Jusqu'à, Instruction, Rien, 68020)

Tracer de manière automatique et paramétrable.

Trace **J)usqu'à**: permet de lancer l'exécution (en mode trace) en évaluant une expression à chaque instruction tracée. L'exécution est interrompue si l'évaluation est vraie (-1).

## [T] J)usqu'à d0!=0

Lance l'exécution en mode trace jusqu'à ce que d0 soit différent de 0 (zéro).

## Trace I)nstruction:

permet de lancer l'exécution (en mode trace) en comparant à chaque instruction la prochaine instruction avec l'instruction définie au départ.

En cas de succès de la comparaison, l'exécution s'arrête.

# **[T] I)nstruction** NOP

Lance l'exécution en mode trace jusqu'à ce que la prochaine instruction soit NOP.

## Trace R)ien

Lance l'exécution en mode trace.

# [T] R)ien

Lance l'exécution en mode trace.

## Trace **6)8020**

Émule le mode trace **T1** disponible à partir du 68020. Il permet de lancer l'exécution jusqu'à une instruction provoquant un changement de flux du programme.

## √ [T] 6)8020

Lance l'exécution en mode trace jusqu'à un changement de flux du programme.

Instructions provoquant toujours l'arrêt: JMP, JSR, BRA, BSR, RTD, RTE, RTS, RTR.

Instructions conditionnées par la valeur du **CCR**: Bcc.

Instruction conditionnée par la valeur du **CCR** et du registre de donnée concerné: DBcc.

## La sélection de l'option

## Voir l'écran logique Y)es N)o

permet de suivre sur l'écran logique l'évolution de l'exécution. Dans l'autre cas, l'écran visualisé est celui d'Adebug, remis à jour à chaque instruction tracée.

Toutes les options de Trace précitées peuvent être interrompues en pressant les deux touches **[Sft]** simultanément, quel que soit le mode de visualisation.

## Remarques:

Il est préférable d'avoir l'**IPL** d'Adebug identique à celui du programme en cours de débogage si l'on souhaite un arrêt immédiat du programme avec **[Sft\_Sft]**.

# [U]ntil

Placer un **point d'arrêt** à l'adresse demandée et lancer le programme.

# [U] Lancer jusqu'à: ma routine

Lance l'exécution jusqu'à l'adresse de la variable ma\_routine.

# [Ctl\_U]ntil

Placer un **point d'arrêt** à l'adresse de la **fenêtre courante** et lancer le programme.

# [J]ump

Mettre le **PC** à la valeur demandée.

## [J] Aller à: 3f342

Met le **PC** à l'adresse \$3f342.

# [Ctl\_J]ump

Mettre le PC à l'adresse de la fenêtre courante.

# [Ctl T]race

Tracer en survolant les instructions de branchement BSR et JSR.

# [Ctl\_F]orce

Forcer le branchement en cas de saut conditionnel non pris.

# [Ctl\_eX]écute

Placer un **point d'arrêt** à l'instruction suivant le dernier BSR ou JSR effectué.

# [Alt\_eX]écute

Revenir directement à l'instruction suivant le dernier BSR ou JSR effectué.

(Utile pour revenir en cas d'entrée par erreur.)

# [Ctl\_Alt\_J]sr to subroutine

Exécute une routine (devant se terminer par **rts**), dans le mode utilisateur du 68xxx.

# [Ctl\_Alt\_Z] Trace T1

Remplace la combinaison [T] 6)8020 (plus rapide!).

# [Sft\_Ctl\_Alt\_J]sr to subroutine

Exécute une routine (devant se terminer par **rts**), dans le mode super-utilisateur du 68xxx.

# [Sft Alt Help]

Interrompre le programme en cours d'exécution et reprendre la main sous Adebug.

Sette fonction utilisant une routine exécutée par l'interruption VBL, il faut que la Vbl du système soit appelée. Cela signifie que l'**IPL** du programme tracé doit être inférieur à 4.

# [Ctl\_Alt\_F]

Afficher la dernière stack frame.

On peut contrôler l'affichage de la liste à l'aide des touches suivantes:

```
[Flèche bas] ou [>]:
                                      descendre d'une ligne.
[Flèche haut] ou [<]:
                                      remonter d'une ligne.
[Sft_Flèche bas] ou [Espace]:
                                      descendre d'une page.
[Sft_Flèche haut] ou [Sft_Espace]:
                                      remonter d'une page.
[Home]:
                                      amène en début de liste.
                                      amène en fin de liste.
[Sft_Home]:
                                      impression de la liste.
[P]:
[S]:
                                      sauvegarde de la liste.
```

Bus error.

Figure 11: visualisation d'une stack frame ([Ctl\_Alt\_F]).

### 3.11 Points d'arrêt

### [Ctl\_B]reakpoint

Mettre ou enlever un point d'arrêt à l'adresse de la fenêtre courante si elle est en type désassemblage ou source.

Le message

#### Point d'arrêt numéro x mis.

apparaît.

Si l'adresse de la fenêtre est impaire, son contenu non inscriptible, ou s'il n'y a plus de point d'arrêt disponible, le message

### Impossible de mettre le point d'arrêt.

apparaît.

Le numéro du vecteur du point d'arrêt ainsi mis est défini par **VEC** # dans les **préférences**. A chaque point d'arrêt mis, le **vecteur** correspondant est réinstallé. L'expression à évaluer est toujours -1 (moins un) qui signifie **vrai**, ce qui arrête l'exécution si elle parvient au point d'arrêt. Le point d'arrêt mis est toujours simple (non permanent).

Cette commande agit à la manière d'une bascule, c.à.d. qu'un appui place le point d'arrêt et un deuxième appui sur la même ligne l'enlève.

### [Alt\_B]reakpoint

Mettre ou enlever un **point d'arrêt** de manière plus évoluée que la fonction précédente. La syntaxe est:

<adresse>,<expression>,<permanent>,<numéro de vecteur>

Chacun des paramètres peut être omis, mais il en faut au moins un (toujours pour éviter les erreurs).

<adresse> est l'adresse à laquelle sera placé le point d'arrêt. Elle est soumise aux mêmes conditions qu'à la fonction précédente (paire et inscriptible).

Si elle est omise, l'adresse prise est l'adresse de la fenêtre courante.

<expression> sera évaluée à l'atteinte du point d'arrêt. Si l'évaluation renvoie **vrai** (-1), l'exécution s'arrête. Par défaut, l'expression prise est -1 (moins un).

<permanent> vaut 0 si le point d'arrêt ne doit pas être permanent (il sera enlevé si <expression> est vrai (-1)). Il vaut 1 si le point d'arrêt doit être permanent (le point d'arrêt ne peut être enlevé qu'à la main avec [Ctl\_B], [Alt\_B] ou [Ctl\_K]).

Par défaut, <permanent> vaut 0 (zéro).

<numéro de vecteur> est le numéro de vecteur du point d'arrêt. Par défaut, **VEC** # est pris.

### [Alt\_B] Point d'arrêt: pc,d0==0,1,4

Met un point d'arrêt permanent à l'adresse contenue dans le PC, et dont l'évaluation est "d0==0".

Le vecteur d'exception lui correspondant est 4, donc l'instruction utilisée est ILLEGAL.

L'exécution ne s'arrêtera que si d0 vaut 0, et le point d'arrêt ne sera pas ôté.

Pour omettre un ou plusieurs paramètres, il suffit de placer uniquement les virgules correspondantes.

### [Alt\_B] Point d'arrêt: ,,1

Place un point d'arrêt permanent à l'adresse de la fenêtre courante.

### [Ctl\_K]ill

Permet soit d'effacer tous les **points d'arrêt** indépendamment de leur type, expression, permanence et numéro (cet effacement est réalisé automatiquement en cas de fin du programme en cours de débogage ou d'Adebug), soit d'effacer un point d'arrêt particulier (il faut alors répondre non à la question "Détruire tous les points d'arrêt", et donner ensuite le numéro du point d'arrêt à enlever).

### [Ctl\_E]xception

Installer une ou toutes les **exceptions**.

Si l'on ne désire pas installer toutes les exceptions, un numéro d'exception sera demandé jusqu'à la sortie de la ligne de commande (par [Esc]).

### [Ctl\_D]étourne

Détourne une exception **TRAP** et permet ainsi d'arrêter l'exécution en cas d'appel correspondant au numéro donné en premier paramètre. Il est possible de spécifier comme second paramètre le numéro de fonction pour lequel l'exécution doit être arrêtée.

Pour enlever le détournement, il suffit de refaire la même commande.

Le nom de la fonction système est aussi directement reconnu:

€>

#### Malloc

(Les noms des fonctions Gemdos, Bios, Xbios commencent par une majuscule. Les noms des fonctions GEM commencent par une minuscule).

Plusieurs détournements sont possibles: par même numéro de trap, ou bien plusieurs traps, ou bien encore une combinaison des deux.)

Le numéro de fonction considéré est pris en **mot**, comme c'est le cas actuellement pour les fonctions système de l'ATARI.

Le **trap #2** est géré spécialement, et permet de s'arrêter sur toute fonction **AES** ou **VDI**.

appl\_init v\_opnvwk

1) la liste des fonctions AES et VDI peut être visualisée avec la fonction [Ctl\_Alt\_D].

2) certains appels sont rassemblés sous le même nom (leur numéro étant identique, il est impossible de les différencier):

Original	Nouveau
vrq_locator/vsm_locator	v_locator
vrq_valuator/vsm_valuator	v_valuator
vrq_choice/vsm_choice	v_choice
vrq_string/vsm_string	v_string

# [Ctl\_D] Détourner < trap n°[, fonction n°]>: 1,3d Détourne la fonction \$3d (Fopen) du trap #1 (Gemdos).

### [Ctl\_Alt\_D]

Afficher la liste des fonctions **AES** et **VDI** reconnues par la fonction **[Ctl\_D]** et affichées dans le désassemblage.

### [Ctl Alt B]

Afficher la liste des points d'arrêt.

Chacun de ceux-ci se présente comme suit:

<n°>, <type>, <n° de vecteur>, <adresse>, <instruction>, <[n°:expression]>.

<n°> est le numéro du point d'arrêt.

<type> peut être **P** (point d'arrêt Permanent), **R** (point d'arrêt en ROM) ou **S** (point d'arrêt simple).

<n° de vecteur> est le numéro de vecteur de l'exception servant à déclencher le point d'arrêt.

<adresse> est l'adresse du point d'arrêt.

<instruction> est l'instruction sur laquelle se trouve le point d'arrêt.

<[expression]> est l'expression du point d'arrêt (optionnelle).

Liste des points d'arrêt (classés par numéro):

Les touches suivantes sont accessibles dans ce mode:

[Flèche bas] ou [>]	Descendre d'une ligne.
[Flèche haut] ou [<]	Remonter d'une ligne.
[Sft_Flèche bas] ou [Espace]	Descendre d'une page.
[Sft_Flèche haut] ou [Sft_Espace]	Remonter d'une page.
[Home]	Aller en début de liste.
[Sft_Home]	Aller en fin de liste.
[E]	Evaluer une expression
[P]	Imprimer la liste.
[S]	Sauver la liste en ASCII.

```
# T V Info
1 S 20 00100F2E MOVE.L A3,D0 [1:a3==0] Avec [Alt_B]
2 S 20 00100F3A CLR.W D0 [2:-1]
3 T 21 Will catch function $48 (Malloc) of Trap $1 (Gemdos) Avec
4 T 2E Will catch all functions of Trap $E (XBios) [Ctl_D]
6 P 20 00100F88 MOVEA.L D0,A1 [5:-1]
6 P 20 00100F9E MOVE.L A3,-(A7) [P6:{a3}.w==$1234]
7 S 20 00100FB4 MOVE.B (A0),D1 [7:-1]
8 S 20 00100FD4 CLR.B (A1) [8:-1] Avec [Ctl_B]
9 S 20 00101000 DBEQ D1,$100FFA [9:-1]
```

Figure 12: liste des points d'arrêt ([Ctl\_Alt\_B]).

### 3.12 Commandes diverses

### [K]eep

Mémoriser la valeur de tous les **registres du 68xxx** afin de les restituer avec **[R]**estore.

### [R]estore

Restaurer la valeur de tous les **registres du 68xxx** (si ceux-ci ont été sauvés avec **[K]**eep), et rafraîchit l'écran.

La combinaison des fonctions **[K]** et **[R]** est fort utile pour faire plusieurs tests sur une routine précise, par exemple.

### [I]nterrupt Priority Level

Modifier l'IPL interne d'Adebug (0~7).

S'il est supérieur à 3, Adebug utilise alors sa propre gestion clavier.

Ce mode est très utile pour tracer une routine en **interruption**, faire de la programmation système (timers ...), etc.

### [I] IPL interne: 7

Passe en IPL 7 (aucune interruption n'est possible).

[I] IPL interne: 3

Repasse en IPL 3 (état "habituel").

### [Alt\_Help]Impression d'écran

Impression système de l'écran, qui marche même en IPL>5.

Il est conseillé de ne pas être en inverse vidéo pour imprimer l'écran. Au besoin, utilisez **[Ct1\_Alt\_I]**.

L'impression se fait en appelant la fonction de copie d'écran standard située en {\$502}.

### [Alt M]emory free

Afficher la **mémoire système libre** en nombre d'octets (mémoire lente, mémoire rapide et total des deux (si mémoire rapide).

### [H]istorique

Historique des dernières instructions tracées.

Leur nombre maximum est définissable dans les préférences sous la dénomination **STO** #. Elles sont triées par ordre d'exécution, le numéro le plus élevé correspondant à la dernière instruction tracée.

On peut contrôler l'affichage de la liste à l'aide des touches suivantes:

[Flèche bas] ou [>]: descendre d'une ligne.

[Flèche haut] ou [<]: remonter d'une ligne.

[Sft\_Flèche bas] ou [Espace]: descendre d'une page.

[Sft\_Flèche haut] ou [Sft\_Espace]: remonter d'une page.

[Home]: amener à la dernière instruction tracée.

[P]: impression de la liste.

[S]: sauvegarde de la liste.

Figure 13: historique ([H]).

### [F1]

Poser une marque (mémorise l'adresse de la fenêtre courante).

### [F2]

Échanger l'adresse de la marque avec l'adresse de la fenêtre courante.

## [F5]

Relâcher les touches Sft, Ctl et Alt.

### [Alt\_P]rint

Imprimer le contenu de la fenêtre courante.

### [P]rint

**Désassembler** sur **imprimante** ou **disque** une partie de la mémoire avec possibilité de création automatique d'étiquettes. Format: source, longueur, adresse d'une zone mémoire inutilisée pour la création des étiquettes, nom du fichier.

### [P]

Désassembler <début,longueur>:{4},10Adresse des étiquettes <adresse>:100000Sortie sur D)isque I)mprimanteD)Sauve <nom de fichier>:rom.s

Désassemble 16 octets de l'adresse contenue dans \$4 (début de ROM) sur disque sous le nom ROM.S.

La mémoire à partir de \$100000 sera utilisée en tant que tampon intermédiaire pour la génération des étiquettes (labels).

### [E]value

Évaluer une expression.

Affiche le résultat (toujours un long mot) sous forme hexadécimale, décimale, binaire et ASCII.

Voir l'appendice sur l'évaluateur.

### **[E] Expression <e>: 2+2**

Demande le calcul et l'affichage du résultat de 2+2.

## [Alt\_@]

Message de copyright.

### [Ctl\_Alt\_R]

Permet de redémarrer Adebug en lui donnant une adresse absolue comme début de zone d'allocation mémoire.

L'écran d'Adebug sera alloué dans cette zone, donc il ne faut pas qu'elle soit en mémoire rapide.

## [O]utput

### R)S232 E)cran

Passer l'affichage des fenêtres d'Adebug sur l'écran de l'Atari à un affichage sur le **terminal** déporté (Minitel par exemple) et inversement. L'option **R)S232** demande l'affichage sur le terminal, l'option **E)cran** demande l'affichage sur l'écran de l'Atari.

Si rien ne s'affiche sur le terminal, commencez par repasser l'affichage sur l'écran de l'Atari ([O] puis [E])

Vérifiez les connections du terminal à l'Atari. Si le terminal est un Minitel, vérifiez qu'il est en mode **videotexte**. Au besoin, éteignez-le, puis rallumez-le. Vérifiez aussi la **vitesse** et la **parité du port série** dans les **préférences**.

Si rien n'y fait, vérifiez que votre câble supporte bien la vitesse demandée.

### [Ctl\_Alt\_Del]

Reset à chaud (conservation du contenu de la mémoire).

### [Sft\_Ctl\_Alt\_Del]

**Reset** à froid (effacement de toute la mémoire).

## 3.13 Profiler (échantillonneur de temps d'exécution)

Adebug dispose d'un échantillonneur de temps d'exécution intégré, c'est-à-dire de la possibilité de savoir quelle est la proportion de temps machine utilisée par chaque routine.

En effet, lors de l'exécution du programme chargé, Adebug recherche à chaque interruption Vbl le premier symbole précédant le PC au moment de l'interruption. A ce symbole est associé un **compteur**, qui sera incrémenté. Les valeurs des compteurs de chaque variable sont visualisables par la commande [L].

Cette fonctionnalité est active ou non suivant l'état d'un drapeau des **préférences**. Toutefois, la recherche du symbole à chaque Vbl utilise un temps machine négligeable, de sorte qu'il ne vaut pas la peine de se priver de cette possibilité.

Par ailleurs, un autre drapeau des préférences permet d'indiquer à Adebug s'il doit échantillonner le temps uniquement pour les variables **appartenant au programme chargé**, ou bien pour **toutes** les variables.

El Pour que l'échantillonnage de temps puisse se faire, il est indispensable que la Vbl du système soit appelée lorsque le programme débogué est en cours d'exécution (voir la fonction [Sft\_Alt\_Help] qui requiert la même condition).

### [Alt\_R]emettre à zéro le profiler

Remettre à zéro les compteurs associés à chaque variable.

## **APPENDICES**

# Appendice I - Connaissances nécessaires à l'usage d'un débogueur

### Le microprocesseur

Un microprocesseur, c'est quoi? Littéralement, il s'agit d'un petit objet qui permet l'exécution séquentielle d'une série d'instructions. Cela semble évident, mais de cette définition découle un ensemble de notions qui sont indispensables à la compréhension de l'assembleur et donc à l'utilisation bien comprise d'Adebug.

En premier lieu, qu'est-ce qu'une instruction? Quelles notions un circuit électronique, aussi puissant soit-il, peut-il "comprendre"? Car comment exécuter un ordre qu'on ne comprend pas? On peut répondre qu'en l'état actuel de la technologie, un microprocesseur est capable de comprendre, et donc d'exécuter, la plupart des opérations arithmétiques de base (addition et soustraction, multiplication et division pour les plus puissants) ainsi que certaines opérations logiques binaires (nous verrons plus loin ce dont il s'agit.). Il sait également où aller chercher les données sur lesquelles il devra faire ces opérations, où stocker leur résultat et où aller chercher les ordres suivants. C'est à peu près tout, et c'est pourtant très suffisant.

Nous venons de voir que le microprocesseur devait connaître un emplacement dans lequel étaient stockées des données. C'est la **mémoire**, qui sert aussi à ranger le résultat des opérations ainsi que les instructions à suivre. Cette mémoire peut être de deux types: **mémoire vive** (ou RAM) dans laquelle on peut soit lire, soit écrire des données et des instructions mais dont le contenu est perdu dès que le courant est coupé, **mémoire morte** (ou ROM) dont le contenu n'est jamais effacé mais dans laquelle il est impossible d'écrire et qui, de ce fait, ne peut servir qu'à lire des instructions ou des données qui seront toujours les mêmes. Le microprocesseur est en liaison perpétuelle avec ces deux types de mémoire par l'intermédiaire d'un (ou de plusieurs) **bus** par lesquels transitent informations et instructions.

Pour aller chercher une information, le microprocesseur doit connaître l'emplacement de celle-ci dans la mémoire. emplacement se nomme l'adresse et transite par le bus du même nom. Ainsi un microprocesseur comme le 68000 qui équipe l'Atari et qui peut adresser 16 mégaoctets a un bus d'adresse de 24 bits (puisqu'il faut 24 chiffres binaires pour écrire le nombre 16 millions), chaque octet de la mémoire ayant une adresse comprise entre 0 et 16777215. En résumé, le microprocesseur demande à la mémoire par l'intermédiaire du bus d'adresse de lui retourner une donnée située à une adresse précise, donnée qui lui est renvoyée par l'intermédiaire du bus de donnée. Il traite alors la donnée selon l'instruction en cours, puis, s'il lui en est donné l'ordre, stocke le résultat à une autre adresse en transitant par les bus avant d'aller chercher l'instruction suivante à l'adresse qui lui est précisée par un registre interne (le PC ou "Program Counter"), etc...

Le microprocesseur, en plus de cette mémoire externe, dispose en interne de quelques emplacements de stockage temporaire dits **registres**.

Ceux-ci, plus ou moins spécialisés, permettent au microprocesseur d'éviter dans certains cas l'aller et retour des données par les bus, opération qui prend la majeure partie du temps de traitement. En effet, supposons qu'on désire faire l'opération suivante:

# Stocker à l'adresse X le résultat de la division par 10 de la donnée située à l'adresse Y+20.

Sans registres, il faudrait que le microprocesseur aille chercher la donnée à l'adresse Y, qu'il la divise par 10, puis qu'il la stocke à l'adresse X, qu'il aille chercher le résultat qu'il vient de calculer pour faire la seconde opération, et qu'enfin il stocke définitivement le résultat. Il est bien plus simple de conserver le résultat intermédiaire dans un **registre** interne pour y traiter toutes les phases du calcul avant de renvoyer le résultat en mémoire centrale. Dans un 68xxx, qu'il existe 8 registres de ce type qui permettent de stocker chacun 4 octets (soit 32 bits), dits **registres de donnée**, et nommés de D0 à D7; ainsi que 7 registres dit **registres d'adresse** qui au lieu de données permettent de stocker l'adresse, par exemple, d'un tableau de données, et nommés de A0 à A6 (eux aussi d'une taille de 32 bits).

Pour en finir avec les notions qui découlent de la définition du microprocesseur, il reste à traiter le mot **séquentiel**. Un microprocesseur classique (c'est à dire dont l'architecture n'est ni de type réseau , ni de type parallèle, nous verrons cela avec les versions d'Adebug qui fonctionneront sur les ordinateurs du futur

...) traite les instructions les unes après les autres. Cette succession d'instructions s'appelle le **programme**. Le programme, comme les données, est stocké en mémoire, mais le programme interne de l'ordinateur (qu'on appelle le **système d'exploitation**) est plus couramment situé en ROM. Les instructions transitent donc, comme n'importe quelle donnée, par le bus de données. Mais elles ont la particularité de se trouver à l'adresse contenue dans un registre interne particulier du microprocesseur, le **PC** ("Program Counter" ou Compteur Ordinal).

Dès qu'une instruction est en cours d'exécution, le PC est augmenté de la taille de l'instruction de façon à contenir l'adresse de la prochaine instruction. Ce processus est automatique et commence avec la mise sous tension de l'ordinateur pour finir avec son extinction (pour le 68xxx, la première instruction exécutée à la mise sous tension est celle qui se trouve à l'adresse contenue à l'adresse \$4). Seules certaines instructions qui modifient directement le contenu du PC permettent de sauter d'une partie du programme à une autre, ce qui permet, par exemple, d'agir de différente façons en fonction du résultat d'un calcul.

Agir de différentes façons en fonction du résultat d'un calcul? Il faut donc qu'on puisse tester si ce résultat est égal ou non à une valeur donnée. Dans ce but, il existe un autre registre interne dans le microprocesseur, le **CCR** ("Condition Code Register" ou Registre de Code de Condition). Tout calcul, toute comparaison agit sur les bits de ce registre. Quand, par exemple, un calcul induit un résultat nul, un bit spécifique du CCR passe à 1 (le bit Z pour "Zéro"). Mais si le résultat n'est pas nul, ce bit passe à 0. De même, un bit est associé à la retenue produite ou non à la dernière opération (bit C pour "Carry"), un autre est positionné si le résultat est positif ou négatif (bit N pour "Négatif")... Cet ensemble de bits permet dans la plupart des cas de choisir l'embranchement vers lequel le programme va se diriger, un ensemble d'instructions de branchement permettant cette modification du PC en fonction du CCR.

Pour en finir avec ce qu'on nomme l'**architecture** d'un microprocesseur, il reste à définir la notion la plus complexe: la **pile**.

Pour cela il va nous falloir introduire une nouvelle notion: les **sous-programmes**.

Dans un programme, il existe des opérations qui seront exécutées plus souvent que d'autres. Par exemple, si le programme affiche quelque chose sur l'écran, il sera nécessaire de répéter un grand

nombre de fois la suite d'opérations qui affichera un symbole. Il paraît donc plus simple d'écrire une fois pour toutes cette succession d'opérations (dite sous-programme) et de l'exécuter à chaque fois que cela sera nécessaire. On dispose déjà d'instructions qui vont mettre dans le PC l'adresse de ce sousprogramme (on dit couramment appeler un sous-programme), mais comment revenir du sous-programme à l'endroit d'où on l'a appelé puisque cela peut être de n'importe quel endroit du programme principal (par opposition au sous-programme)? La solution est évidemment de stocker (dans la mémoire) l'adresse qui sera mise dans le PC lorsqu'on rencontrera l'instruction de fin de sous-programme. Mais si un sous-programme appelle à son tour un autre sous-programme, et ainsi de suite ? La pile répond à ce besoin: un registre spécial du microprocesseur (nommé A7 ou SP dans le 68xxx) contient l'adresse de la mémoire dans laquelle le microprocesseur aura stocké l'adresse qui devra être mise dans le PC à la fin du sous-programme courant. Si un nouveau sous-programme est appelé avant la fin du sousprogramme courant, le contenu de A7 sera diminué (de 4 octets, place nécessaire pour stocker une adresse) de façon à pointer sur une case mémoire au-dessus de celle dans laquelle on avait mis l'adresse de retour du sous-programme précédent et ainsi de suite. Il s'ensuit qu'une partie de la mémoire, dite pile, doit être pour stocker toutes ces réservée adresses. Ouand microprocesseur rencontre une instruction de fin de sous programme, il va mettre dans le PC, le contenu de l'adresse pointée par A7, puis ajoutera 4 à A7 de façon à ce que ce registre pointe toujours vers l'adresse de retour du sous-programme en cours d'exécution. La dernière adresse stockée dans la pile étant la première à être relue par le microprocesseur quand il rencontre une instruction de retour. On appelle cela une structure LIFO, ("Last In, First Out" ou dernier entré, premier sorti).

On vient de voir qu'un microprocesseur ne faisait qu'une chose à la fois. Pourtant, pendant l'exécution d'un programme, la souris peut toujours être déplacée, et les touches qu'on tape au clavier sont bien prises en compte. Cela est dû aux **interruptions**. Sans entrer dans le détail, disons simplement qu'une interruption est une forme particulière de sous-programme qui est appelé non à la demande du programme en cours, mais à la demande d'un **périphérique** (par exemple clavier ou lecteur de disquette) ou d'un **timer** qui permet au microprocesseur d'exécuter certaines tâches répétitives comme, par exemple, la gestion du clavier pour voir si celui-ci a été actionné et pour agir en conséquence. Dans l'Atari, on compte au moins 4 timers de ce genre.

Enfin et pour être tout à fait complet, sachez que le 68xxx peut travailler dans 2 modes. Le mode **utilisateur** qui interdit au programme en cours d'exécuter certaines instructions dites "privilégiées" (comme RTE ou MOVE to SR), et le mode **superviseur** qui n'est pas limité et qui dispose de son propre registre de pile (nommé A7' ou SSP). Les interruptions travaillent toujours en mode superviseur.

# L'assembleur

## Les modes d'adressage

Il existe plusieurs façons d'indiquer au microprocesseur l'adresse à laquelle se trouve la donnée que l'instruction en cours doit traiter (ou **opérande**) ainsi que l'adresse à laquelle le résultat du traitement sera stocké (ou **destination**). C'est ce qu'on appelle les **modes d'adressage**.

Le 68000 dispose de 12 modes d'adressage.

### L'adressage immédiat

L'opérande est compris dans le corps de l'instruction. Le processeur n'a donc pas besoin d'en connaître l'adresse.

MOVE.W #\$1B,D0 permet de charger **immédiatement** la valeur hexadécimale \$1B dans les 16 bits de poids faible du registre D0.

### L'adressage direct d'un registre de donnée

L'adresse de la donnée n'est pas en mémoire, mais directement dans un des registres de donnée du microprocesseur.

Dans l'exemple précédent, on a stocké **directement** la valeur hexadécimale \$1B dans le registre D0.

### L'adressage direct d'un registre d'adresse

Comme le précédent mais pour un registre d'adresse du 68000.

- ✓ MOVE.L #\$1234,A1 stocke dans les 32 bits du registre A1 la valeur \$1234.
- Les opérations sur les registres d'adresse sont automatiquement étendues.
- MOVE.W #\$1234,A1 donne le même résultat que ci-dessus, en utilisant 2 octets de moins.

En revanche:

MOVE.W #\$8001,A1 stocke dans les 32 bits de A1 la valeur \$FFFF8001.

## L'adressage indirect d'un registre d'adresse

L'adresse de destination est en mémoire mais elle est pointée par la valeur d'un registre d'adresse.

MOVE.B #\$1B,(A0) stocke à l'adresse pointée par A0 l'octet \$1B.

### L'adressage indirect d'un registre d'adresse avec postincrémentation

Comme le précédent mais la valeur du registre d'adresse est augmentée après l'exécution de la taille de l'opérande (de 1,2 ou 4 octets).

✓ MOVE.W #\$1B,(A0)+ stocke à l'adresse pointée par A0 le mot \$1B puis ajoute 2 à A0.

### L'adressage indirect d'un registre d'adresse avec prédécrémentation

On soustrait à la valeur du registre d'adresse la taille de l'opérande, puis on stocke l'opérande à la nouvelle adresse pointée par le registre. C'est ce mode qu'utilise le 68000 pour sa gestion de pile.

✓ MOVE.L #\$1B,-(A7) soustrait 4 à la valeur de A7, puis stocke le long mot \$1B à l'adresse ainsi calculée.

# L'adressage indirect d'un registre d'adresse avec déplacement

L'adresse est calculée à partir d'un des registres d'adresse, mais en y ajoutant (ou soustrayant) une valeur de déplacement sur 16 bits signés (donc de -32768 à +32767).

# L'adressage indirect d'un registre d'adresse avec index et déplacement

L'adresse est obtenue en ajoutant à la valeur contenue dans le registre d'adresse, le nombre signé contenu dans le registre servant d'index ainsi que le déplacement codé sur 8 bits signés. Le registre servant d'index peut être soit un registre de donnée, soit comme c'est moins connu un autre registre d'adresse. Sa valeur peut être codée sur 16 ou sur 32 bits au choix du programmeur. A partir du 68020, un nombre (1, 2, 4 ou 8) peut être dynamiquement multiplié à la valeur de l'index.

- MOVE.W \$42(A1,D5.L),D0 stocke dans le registre D0 le mot contenu à l'adresse pointée par A1 plus #\$42 plus les 16 bits de poids faible du registre D5.
- MOVE.W \$42(A1,D5.L\*2),D0 est identique à l'exemple précédent, mais multiplie par 2 la valeur de D5 avant de l'ajouter au reste.

# L'adressage indirect d'un registre avec index et déplacement sur 32 bits

Comme ci-dessus, mais autorise un déplacement sur 16 ou 32 bits. De plus, chaque élément est supprimable.

- ✓ MOVE.W (\$12345678,A0,D2\*4),D0 stocke dans D0 le mot contenu à l'adresse \$12345678+A0+D2\*4.
- ✓ MOVE.W (\$12345678,D2\*4),D0 stocke dans D0 le mot contenu à l'adresse \$12345678+D2\*4.
- Mode d'adressage disponible à partir du 68020.

# L'adressage en double indirection d'un registre avec index et déplacement sur 32 bits

Comme ci-dessus, mais permet un niveau d'indirection et un déplacement supplémentaires.

- ✓ MOVE.W ([\$12345678,A0,D2\*4]),D0 stocke dans D0 le mot contenu à l'adresse située en \$12345678+A0+D2\*4.
- ✓ MOVE.W ([\$12345678,a0],D2\*4,\$1234),D0 stocke dans D0 le mot contenu à l'adresse \$12345678+A0 additionné de D2\*4+\$1234.

Mode d'adressage disponible à partir du 68020.

### L'adressage absolu court

C'est le pendant de l'adressage immédiat. Si celui-ci concerne l'opérande, celui-là concerne l'adresse. Elle est indiquée dans le corps du programme par un mot dont le processeur étend le bit de signe (voir instruction EXT) avant d'aller y chercher une donnée.

✓ MOVE.L \$44E.W,D0 stocke dans D0 le long mot contenu à l'adresse \$44E.

### L'adressage absolu long

Comme le précédent mais l'adresse est indiquée dans le corps du programme par un long mot. Il vaut mieux dans ce cas utiliser ce dernier.

MOVE.W \$F8000,D0 stocke dans D0 le mot contenu à l'adresse \$F8000.

### L'adressage relatif au PC avec déplacement

L'adresse de la donnée est obtenue en ajoutant un déplacement sur 16 bits signés à l'adresse contenue dans le PC. Au moment ou ce calcul est effectué, le PC pointe non pas sur l'instruction en cours de traitement, mais sur le mot qui signale ce mode d'adressage dans le corps du programme.

✓ MOVE.W \$442(PC),D1 stocke dans D1, si cette instruction est située à l'adresse #\$50000 de la mémoire, le mot qui se trouve en \$50002+\$442.

### L'adressage relatif au PC avec index et déplacement

Comme le précédent mais le calcul dépend en plus d'un registre d'index sur 16 ou 32 bits.

✓ MOVE.B \$42(PC,D0.L),D1 stocke dans D1 l'octet qui se trouve en PC+2+#\$42+D0.

### L'adressage indirect du PC avec index et déplacement sur 32 bits

Comme ci-dessus, mais autorise un déplacement sur 16 ou 32 bits. De plus, chaque élément est supprimable.

MOVE.W (\$12345678,PC,D2\*4),D0 stocke dans D0 le mot

contenu à l'adresse \$12345678+4+PC+D2\*4.

- MOVE.W (\$12345678,ZPC,D2\*4),D0 stocke dans D0 le mot contenu à l'adresse \$12345678+4+D2\*4. (ZPC signifie PC supprimé).
- Mode d'adressage disponible à partir du 68020.

# L'adressage en double indirection du PC avec index et déplacement sur 32 bits

Comme ci-dessus, mais permet un niveau d'indirection et un déplacement supplémentaires.

- ✓ MOVE.W ([\$12345678,PC,D2\*4]),D0 stocke dans D0 le mot contenu à l'adresse située en \$12345678+4+PC+D2\*4.
- ✓ MOVE.W ([\$12345678,PC],D2\*4,\$1234),D0 stocke dans D0 le
  mot contenu à l'adresse \$12345678+4+PC additionné de
  D2\*4+\$1234.
- Mode d'adressage disponible à partir du 68020.
- Ces quatre derniers modes d'adressage permettent la génération d'un programme n'ayant pas besoin d'être relogé.

### Les instructions du 68xxx

On a déjà évoqué quelques-unes des instructions du langage-machine, mais tout langage informatique possède les instructions minimum qui permettent le **traitement de l'information**. On peut les classer dans trois grandes familles: instructions de **traitement numérique** (arithmétique et logique), instructions d'entrées/sorties et instructions de **tests et branchements**.

instructions de traitement arithmétique

modacione de d'alternent antimiodique		
ADD	addition (+)	
SUB	soustraction (-)	
CLR	mise à zéro	
NEG	négation (-)	
DIVU.W	division (/) (word/word=word)	
DIVU.L	division (/) (long/long=long ou quad)	
MULU.W	multiplication (*) (word*word=long)	
MULU.L	multiplication (*) (long*long=quad)	
EXT	extension de signe	
SWAP	échange des mots bas et haut d'un long	

instructions de traitement logique

AND	et (&)
EOR	ou exclusif (~)
OR	ou inclusif ( )
NOT	inversion (^)
ASL/ASR	décalage signé vers la gauche/droite
LSL/LSR	décalage non signé vers la
	gauche/droite
ROL/ROR	rotation vers la gauche/droite

instructions de manipulation d'1 bit

BCHG	inverse l'état d'un bit	
BCLR	met à zéro un bit	
BSET	met à un un bit	
BTST	teste l'état d'un bit	

# instructions de manipulation de bits

BFCHG	inverse l'état de plusieurs bits	
BFCLR	met à zéro plusieurs bits	
BFEXT	extrait plusieurs bits	
BFFFO	trouve le premier bit	
BFINS	insère plusieurs bits	
BFSET	met à un plusieurs bits	
BFTST	teste l'état de plusieurs bits	

## instructions d'entrées/sorties

MOVE	met une valeur	
MOVEM	met plusieurs valeurs	
LEA	met une adresse dans un registre d'adresse	
PEA	empile une adresse	
EXG	échange deux valeurs de registres	
MOVEP	met une valeur en prenant un octet sur	
	deux	

### instructions de test

CMP	compare	
TST	teste	
TAS	teste puis met à -1	
Scc	met à 1 si condition vraie (sinon à 0)	
Bcc	branche si condition vraie	
DBcc	teste le registre et branche si condition vraie	
TRAPcc	trap si condition vraie	

## instructions de branchement

BRA	Branche (PC relatif)	
JMP	branche (absolu)	
BSR	branche avec empilement (retour par RTS)	
JSR	idem	
TRAP	exception TRAP (retour par RTE)	
RTD	dépile et revient d'un BSR ou JSR	
RTE	revient d'une interruption	
RTR	dépile le CCR et revient d'un BSR ou JSR	
RTS	revient d'un BSR ou JSR	

## Les instructions diverses

LINK	Empile le registre spécifié et lui donne la	
	valeur courante de la pile	
UNLK	Restaure la pile et le registre empilé	
NOP	Ne fait rien	

## Notions de trace

La première fonction d'un débogueur est de **tracer** un programme. Tracer signifie "exécuter l'instruction courante pour voir ce qui ce passe".

Concrètement, le programme devant être tracé ayant été chargé avec la fonction [Ctl\_L], à la première ligne de la fenêtre 2, un symbole (**TEXT**) est apparu, suivi d'une petite **flèche**. L'instruction indiquée par ces deux repères est en fait la première instruction du programme.

La petite flèche indique toujours l'instruction où en est le **PC**. Pour la tracer (donc pour exécuter l'instruction puis voir ce qui c'est passé), il suffit d'appuyer sur **[Ct1\_Z]** une fois. A la suite de quoi, l'écran ayant été redessiné, le nouvel état des registres ainsi que la nouvelle valeur du **PC** (indiqué par la petite flèche) sont visibles. Ca y est, vous venez de tracer une instruction! Et en continuant de la sorte vous tracerez les instructions suivantes les unes après les autres.

### Le mode Trace un peu plus en détail

Le 68xxx possède un **Status Register** (**SR**) dont presque chaque bit a une signification.

Le SR courant est affiché dans les fenêtres en mode registres, avec sa valeur numérique puis des lettres symbolisant l'état des bits. Le mode **Trace** est indiqué par le positionnement du bit **T** du SR (bit 15), ce qui signale au 68xxx de générer une exception "trace" après chaque instruction exécutée.

Et mode **Trace** correspondant à un vecteur d'exception du 68xxx, dont l'adresse se situe en \$24, tout traçage d'une instruction altérant le contenu de cette adresse risque de provoquer le blocage du système.

# Notions de point d'arrêt / exception

Il est généralement fastidieux de tracer depuis le début du programme jusqu'à la routine à déboguer. Un moyen plus rapide consiste à placer un **point d'arrêt** sur la première instruction de cette routine et de lancer l'exécution du programme (par la fonction **[U]**).

Lorsque le PC arrivera sur le point d'arrêt, la main sera rendue à Adebug et il sera possible de tracer à loisir.

Il est possible de placer un point d'arrêt sans lancer l'exécution, il suffit pour cela d'amener la fenêtre courante à l'adresse désirée puis de faire [Ctl\_B] (la fenêtre doit être de type désassemblage). Ensuite, à droite de l'instruction apparaît [-1] (moins un), ce qui signifie qu'un point d'arrêt dont l'évaluation vaut -1 (point d'arrêt simple) vient d'être posé.

Sur la ligne d'information est alors affiché le message **Point d'arrêt n°x mis.** 

(x étant le numéro du point d'arrêt).

## Les points d'arrêt en détail

Un point d'arrêt correspond en fait à une exception du 68xxx. Lorsque le PC arrive sur l'instruction de point d'arrêt l'exécution de cette dernière provoque une exception, rendant la main au débogueur qui affiche alors

#### Point d'arrêt n°x atteint.

Il est donc possible en théorie d'utiliser tous les vecteurs d'exception comme point d'arrêt. Cependant certains ne sont générables qu'électroniquement, et d'autres dépendent de l'environnement. C'est pourquoi seuls l'**instruction illégale** et les **traps** (sauf les traps 1, 2, 13 et 14 si l'on veut utiliser le système) sont utilisables.

Le vecteur d'exception utilisé par défaut pour un point d'arrêt (lors de [Ctl\_A], [Ctl\_B], [Ctl\_T], [Ctl\_U], [Ctl\_X] ou de [Alt\_B] sans préciser de numéro de vecteur) est dénommé **VEC** #, et se trouve défini dans les préférences. En l'absence du fichier de configuration, il est fixé à \$20=\32, soit Trap #0.

Si en cours d'exécution du programme à déboguer, l'adresse contenue dans le vecteur d'exception du point d'arrêt est modifiée, il y a peu de chances qu'en arrivant sur ce dernier le débogueur reprenne la main de façon correcte.

# Notions d'interruption / exception

Si maintenant, alors que vous êtes en train de lire ce manuel, le téléphone vient à sonner, vous allez sans doute interrompre votre activité pour aller répondre, discuter quelque temps avec votre interlocuteur, puis reprendre votre activité.

Cet exemple reflète assez fidèlement le mécanisme de fonctionnement de ce que l'on nomme **interruption** au niveau d'un processeur tel que le **MC68xxx**. C'est pourquoi nous allons l'approfondir un peu plus.

Plutôt que de décrocher en permanence le téléphone afin de savoir si quelqu'un cherche à vous appeler, vous allez à vos occupations habituelles et lorsqu'un signal, en l'occurrence la sonnerie du téléphone, vous avertira que quelqu'un cherche à vous joindre, alors seulement vous irez décrocher.

Supposons maintenant que vous ayez programmé une application effectuant sans cesse une certaine tâche et que vous souhaitiez qu'elle s'arrête dès la pression d'une touche du clavier. Il vous serait bien sûr possible d'introduire, un peu partout dans le code de votre programme, des tests du clavier. Cependant, cette solution est mauvaise; en effet elle est d'une part lourde à mettre en oeuvre, car elle demande d'insérer des parties de codes dans un programme existant et par conséquent de le modifier, et d'autre part inefficace car si l'utilisateur presse une touche du clavier entre deux tests, et non pendant l'un d'eux, cette action risque de ne pas être prise en compte par le programme.

Par contre, si l'on programme le processeur de telle sorte que l'appui sur l'une des touches du clavier provoque une interruption du programme en cours d'exécution, l'arrêt sera inévitable et instantané dès la première pression.

Cette solution a pour avantage, outre sa fiabilité et sa simplicité de mise en oeuvre, de ne pas consommer de temps machine alors que tester sans cesse le clavier perd un temps qui peut ne pas être négligeable si cette opération est répétée des milliers de fois.

## Gestion d'une interruption par le 68xxx

Au niveau du 68xxx, les interruptions sont gérées comme des sous-programmes.

Si, lors de l'exécution d'un programme, une interruption survient, le 68xxx va sauvegarder son **état** (selon une séquence bien précise que nous décrirons plus loin), aller exécuter un **sous-programme spécifique** au type d'interruption venant de se produire puis restaurer l'état précédemment sauvegardé pour enfin reprendre l'exécution du programme là où il s'était arrêté.

A chaque type d'interruption correspond un **vecteur** placé, à adresse fixe, en début de mémoire (Cf. figure 10) contenant l'adresse de la routine à laquelle va se brancher le 68xxx si l'interruption correspondante survient.

N° Vec	Adresse	Type d'interruption	Commentaire
1	4	Reset	
2	8	Erreur de bus	
3	C	Adresse impaire	
4	10	Instruction Illégale	
5	14	Division par zéro	
6	18	Instruction CHK, CHK2	
7	1C	Instruction TRAPV, TRAPcc	
8	20	Violation de Privilège	
9	24	Trace	
A	28		
В	2C	Ligne A	
		Ligne F	
C D	30	Violeties de systemale even va	
l D	34	Violation de protocole avec un	
	20	coprocesseur	
E	38	Erreur de format	
F	3C	Interruption Non Initialisée	
10-17	40-5C	-	
18	60	Interruption Parasite	
19	64	Autovecteur (IPL1)	
1A	68	Autovecteur (IPL2)	Interruption HBL
1B	6C	Autovecteur (IPL3)	
1C	70	Autovecteur (IPL4)	Interruption VBL
1D	74	Autovecteur (IPL5)	
1E	78	Autovecteur (IPL6)	Interruption MFP
1F	7C	Autovecteur (IPL7)	
20	80	Trap #0	
21	84	Trap #1	Gemdos
22	88	Trap #2	AES/VDI
23-2C	8C-B0	Trap #3-12	
2D	B4	Trap #13	Bios
2E	B8	Trap #14	Xbios
2F	BC	Trap #15	
30	C0	FPCP Branch or Set on	FPU
0.4		Unordered Condition.	EDIT
31	C4	FPCP Inexact Result.	FPU
32	C8	FPCP Divide By Zero.	FPU
33	CC	FPCP Underflow	FPU
34	D0	FPCP Operand Error	FPU
35	D4	FPCP Overflow	FPU
36	D8	FPCP Signaling NAN	FPU
37	DC	-	
38	E0	Erreur de configuration MMU	MMU
39	E4	68851	MMU
3A	E8	68851	MMU
3B-3F	EC-FC	-	
40-7F	100-1FC	Vecteurs utilisateurs	

Figure 14: table des vecteurs d'exceptions

N° Vec	Adresse	Utilisation sur Atari
40	100	Port parallèle (Centronics Busy)
41	104	Port série (Carrier Detect DCD)

42	108	Port série (Clear To Send CTS)
43	10C	Fin de blitter
44	110	Timer D (RS232 Baud Rate Generator USART)
45	114	Timer C (200 Hz)
46	118	Clavier / Midi
47	11C	FDC/HDC
48	120	Timer B (HBL)
49	124	RS232 Transmit Error
4A	128	RS232 Transmit Buffer Empty
4B	12C	RS232 Receive Buffer Empty
4C	130	RS232 Receive Buffer Full
4D	134	Timer A (Son sur STe)
4E	138	RS232 Ring Indicator
4F	13C	Monochrome Detect (GPI7)
50-5F	140-17C	Autovecteurs MFP 2
60	180-184	Zylog port A TX Interrupt
61	188-18C	Zylog port A External Interrupt
62	190-194	Zylog port A RX Interrupt
63	198-19C	Zylog port A Special Interrupt
64	1A0-1A4	Zylog port B TX Interrupt
65	1A8-1AC	Zylog port B External Interrupt
66	1B0-1B4	Zylog port B RX Interrupt
67	1B8-1BC	Zylog port B Special Interrupt
68-7F	1C0-1FC	-

Figure 15: table des vecteurs utilisateurs

### Convention

Certaines interruptions sont provoquées par des instructions du 68xxx (CHK, ILLEGAL, TRAP ...) et sont alors appelées exceptions.

# Prise en charge d'une interruption par le 68xxx

Lorsque le 68xxx s'interrompt pour donner le contrôle à une routine d'interruption, il observe auparavant la séquence suivante (sauf pour l'exception Reset qui ne fait aucune sauvegarde):

- 1. Le SR (Registre d'état) est copié dans un registre temporaire interne.
- 2. Le bit superviseur est positionné et pour les interruptions autres que Division par zéro, CHK, et TRAP le mode **Trace** est désactivé.
- 3. L'adresse de la routine d'interruption est chargée depuis la table des vecteurs.
- 4. Les numéros d'exception et de stack frame sont empilés (à partir du 68010).
- 5. Le PC (Compteur ordinal) est empilé.
- 6. Le SR est empilé.
- 7. Sur le 68000 et pour les exceptions **Erreur de bus** et **Adresse impaire**, le processeur empile également:
  - 7A. Un mot contenant l'instruction ayant provoqué l'erreur.
- 7B. Un mot long contenant l'adresse véhiculée sur le bus lors de l'erreur.
- 7C. Un mot dont seuls les 5 bits les moins significatifs sont utilisés et ont la signification suivante:

7Ca. Bit 0-2 signaux FC0-2.

7Cb. Bit 3 à 1 signifie que l'erreur est survenue en espace CPU.

7Cc. Bit 4 à 0 signifie que l'erreur est survenue pendant un cycle d'écriture, et à 1 qu'elle est survenue pendant un cycle de lecture.

ou

- 7. A partir du 68010, la plupart des exceptions empilent une stack frame particulière.
- 8. Branche à l'adresse précédemment lue dans la table de vecteurs.
- On a coutume de revenir d'une interruption au programme principal par l'instruction RTE qui restaure notamment le SR et le PC.

### Priorité des interruptions

Si, lors de l'exécution d'une instruction, plusieurs interruptions surviennent, le 68xxx les exécute en utilisant une table de priorité.

Priorité la plus forte.

Interruptions provoquant l'arrêt immédiat de la commande en cours:

- -Reset.
- -Adresse impaire.
- -Erreur de bus.

Interruptions faisant partie de l'instruction traitée:

- -Division par zéro.
- -CHK.

Interruptions s'exécutant avant l'instruction pointée par le PC:

- -Instruction illégale.
- -Violation privilège.
- -Ligne A.
- -Ligne F.

Interruptions s'exécutant à la fin de l'instruction pointée par le PC:

- Trace.
- Interruption périphérique.

Priorité la plus faible.

# Appendice II - Périphériques utilisables

## 1 - Ecran

Adebug gère un écran et une résolution qui lui sont propres (**logiques**), et ceux du programme en cours de débogage (**physiques**.

Le passage de l'écran physique à l'écran logique se fait manuellement par la commande [V]. Lors du débogage, il est aussi conditionné par certains **drapeaux** des préférences, ou par un choix dans la commande [T].

# Disque (souple et dur)

Les opérations disque d'Adebug sont celles du système d'exploitation. En conséquence l'accès au disque ne peut s'opérer en IPL supérieur à 5. (on peut toutefois outrepasser cette interdiction avec le drapeau correspondant dans les **préférences**, à ses risques et périls). Le message

# Opérations disque dur non autorisées en IPL>5.

apparaîtra en cas d'erreur.

Les noms de fichiers sont expansés, i.e. les caractères joker standards du système d'exploitation (? et \*) sont pris en compte. Du fait qu'il n'y a pas de souris sous Adebug et qu'il utilise son propre écran physique, il lui est impossible d'utiliser les boites d'alerte d'erreur disque standard. En conséquence, en cas d'erreur disque, Adebug renvoie simplement le type d'erreur après avoir désélectionné le lecteur si nécessaire.

# Le port série (RS232)

Si vous désirez utiliser le mode **terminal** pour déboguer un programme, il vous faut connaître certains détails. La communication entre Adebug et ce terminal se fera en mode asynchrone, tous les autres paramètres étant réglables à votre choix.

Par défaut, Adebug est configuré pour l'utilisation d'un Minitel. Si c'est le cas, Adebug adaptera sa vitesse à votre Minitel automatiquement. Tout ce que vous aurez à faire est de vous

assurer que le Minitel est bien, au départ, en mode Vidéotex (40 colonnes). Tout le reste est automatique et Adebug utilisera la vitesse la plus rapide admise par votre Minitel (c. à. d. 4800 bauds pour un Minitel 1b et 9600 bauds pour un Minitel 2. Si vous n'avez qu'un Minitel 1 ancien modèle, courez vite en changer à votre agence commerciale France Telecom).

Si votre terminal n'est pas un Minitel, voici les différents paramètres qu'il vous faudra choisir dans les préférences:

### Minitel Y)es N)o

Tapez N.

### Vitesse du port série:

Choisissez la vitesse en bauds à laquelle votre terminal est configuré dans cette table:

Valeur	Vitesse (bauds)
0	19200
1	9600
2	4800
3	3600
4	2400
5	2000
6	1800
7	1200

Figure 16: codage du port série

En dessous de 1200 bauds (et encore) il est inutile de vouloir utiliser un terminal déporté, la vitesse d'affichage étant prohibitive. De même, il est conseillé de désactiver l'option **Temps réel** des préférences pour toute vitesse inférieure à 9600 bauds.

### Parité du port série:

Comme son nom de l'indique pas, il vous faut ici choisir non seulement le type de parité utilisé par votre terminal, mais aussi le nombre de bits significatifs et le nombre de bits de stop. Voici quel est le codage de cette valeur:

Bit	à O	à 1
0	-	1
1	parité impaire	parité paire
2	parité ignorée	parité validée
3	-	1
4	1 bit de stop	2 bits de stop
5	8 bits/caractère	7 bits/caractère
6	0	-
7	_	1.

Figure 17: codage de la parité du port série

Si votre terminal est un Minitel mais qu'il ne répond pas (ou mal), vérifiez dans les préférences le choix **Minitel Y)es N)o**. (Tapez Y).

# Le port parallèle (imprimante)

Adebug permet d'imprimer avec ou sans faire appel au système, suivant l'état du drapeau **Impression système** (voir le chapitre sur les **préférences**).

Dans la cas de l'impression non système, Adebug utilise ses propres routines d'impression par le **port parallèle**. De ce fait il est inutile de reconfigurer (via le panneau de contrôle) l'imprimante. Le protocole utilisé est un protocole standard pour imprimante matricielle. Si l'imprimante n'est pas ou plus prête à recevoir de données, le message

### Erreur d'impression.

apparaît. Vous pouvez alors vérifier les connections et recommencer.

# Appendice III - L'évaluateur

Plus que tout autre **débogueur**, Adebug passe une grande partie de son temps à effectuer des calculs, que ce soit dans le cadre de sa gestion du **temps réel**, du mode **trace**, des **points d'arrêt**, etc... Nous avons donc apporté un soin tout particulier à la conception et à la réalisation de son **évaluateur**.

Celui-ci gère les opérations mathématiques et booléennes courantes, les variables, les points d'arrêt, les fenêtres, les registres du 68xxx ainsi que les tests et les affectations.

### Formats de constantes acceptés

L'évaluateur reconnaît les constantes en notation **hexadécimale** (préfixe \$), **décimale** (préfixe \), **octale** (préfixe @) et **binaire** (préfixe %).

Bien sûr, tout calcul avec combinaison de différentes notations est valide.

Toute constante sans préfixe est considérée en notation hexadécimale.

### Opérateurs arithmétiques

Les opérateurs mathématiques standards +, -, \*, / sont acceptés, avec leurs priorités respectives.

```
10-(8/2) (=6) (10-8)/2 (=1) (10-8/2 (=6)
```

car la division a priorité sur l'addition.

De plus, les opérateurs de pré-incrémentation, post-incrémentation, pré-décrémentation, post-décrémentation ++ et -- (comme en langage C) sont également supportés.

soustrait un à la valeur de d0, affecte d0 avec cette valeur, et renvoie la nouvelle valeur de d0.

### Opérateurs logiques

Tous les opérateurs logiques disponibles dans le jeu d'instructions du 68xxx sont reconnus:

Nom	Instruction 68xxx	préfixe
OR exclusif	EOR	^
OR inclusif	OR	I
ET	AND	&
INVERSE	NOT	~
Décalage logique à gauche	LSL	<<
Décalage logique à droite	LSR	>>

### Points d'arrêt

Les **points d'arrêts** sont identifiés grâce à leur numéro (visible avec la fonction **[Ctl\_Alt\_B]**). Le préfixe # (dièse) leur est réservé.

### **Fenêtres**

L'adresse de base d'une fenêtre est accessible par la notation w<chiffre> (w1~w5 pour les fenêtres 1~5).

L'adresse de base de la fenêtre courante est accessible par la notation **cw**.

### Indirections et parenthèses

Les symboles ( et ) signalent un niveau de priorité supérieur, comme en mathématiques classiques.

Exécution des calculs:

5\*4 (=20) -2+20 (=18) 18+4 (=22) 22-5 (=17) 17+8 (=25 comme résultat final)

Les symboles { et } fournissent le contenu de la chaîne englobée (indirection).

Si, au cours d'un calcul d'**indirection**, l'évaluateur rencontre une adresse non lisible, une erreur de syntaxe est indiquée (l'écran clignote).

### Variables, blocs, routines (LA, LR, EQ, EX, RO, BL)

Toute variable, bloc ou routine peut faire l'objet d'un calcul ou d'un test. Elle est alors caractérisée par son nom (visible avec la fonction **[L]**)

#### linea

exécute la routine linea.ro (que vous trouverez sur la disquette d'Adebug).

toto+10 tata=1245678 ({titi}.w+20)=tutu

Seules les variables de type **LA** peuvent être affectées par l'évaluateur. Toute affectation d'un **LR**, **EQ**, **EX**, **BL** ou **RO** renvoie une erreur.

### Passages de paramètres dans les routines

Si la routine appelée a besoin d'un ou de plusieurs paramètres, l'évaluateur est capable de les gérer comme toute autre expression.

Ceux-ci doivent seulement être placés entre parenthèses.

### fface\_bloc(myscreen,\32000)

Les paramètres myscreen (adresse de début d'écran par exemple) et \32000 sont passés à la routine efface\_bloc

Une routine peut recevoir au maximum 50 paramètres.

# Registres du 68xxx

Tous les registres du 68000 sont utilisables pour les calculs:

**DO** à **D7**, **A0** à **A7**, **SP** (pile courante), **USP** (pile utilisateur), **SSP** (pile superviseur), **SR** (registre d'état), **CCR** (registre des codes de condition), **PC** (compteur ordinal).

Les registres 68xxx sont aussi reconnus:

VBR, ISP, MSP, CACR, CAAR, SFC, DFC, TC, CRP, SRP, TTO, TT1, MMUSR, FPO à FP7, FPCR, FPSR, FPIAR.

Les noms de registres peuvent être entrés en majuscule comme en minuscule. Pour éviter toute confusion entre registres et constantes hexadécimales, entrer \$a0 ou 0a0 pour la constante et a0 pour le registre.

L'évaluateur reconnaît directement toutes les entrées-sorties du ST et STe par leur nom, ainsi que leur taille.

memconf renvoie \$ff8001. {memconf}

renvoie l'octet contenu en \$ff8001.

#### Entrées-sorties reconnues:

Nom	Adresse	Taille	Remarque
COLOR0	\$ff8240	WORD	_
COLOR1	\$ff8242	WORD	
COLOR2	\$ff8244	WORD	
COLOR3	\$ff8246	WORD	
COLOR4	\$ff8248	WORD	
COLOR5	\$ff824a	WORD	
COLOR6	\$ff824c	WORD	
COLOR7	\$ff824e	WORD	
COLOR8	\$ff8250	WORD	
COLOR9	\$ff8252	WORD	
COLOR10	\$ff8254	WORD	
COLOR11	\$ff8256	WORD	
COLOR12	\$ff8258	WORD	
COLOR13	\$ff825a	WORD	
COLOR14	\$ff825c	WORD	
COLOR15	\$ff825e	WORD	
MEMCONF	\$ff8001	BYTE	

DDVCLII	¢ff0001	BYTE	
DBASEH	\$ff8201	BYTE	
DBASEM DBASEL	\$ff8203 \$ff820d	BYTE	STe
			Sie
VCOUNTHI	\$ff8205	BYTE	
VCOUNTMID	\$ff8207	BYTE	
VCOUNTLO	\$ff8209	BYTE	
SYNCMODE	\$ff820a	BYTE	
SHIFTMODE	\$ff8260	BYTE	O/D
HSCROLL	\$ff8265	BYTE	STe
LINEWID	\$ff820f	BYTE	STe
HALFTONEO	\$ff8a00	WORD	Blitter
HALFTONE1	\$ff8a02	WORD	Blitter
HALFTONE2	\$ff8a04	WORD	Blitter
HALFTONE3	\$ff8a06	WORD	Blitter
HALFTONE4	\$ff8a08	WORD	Blitter
HALFTONE5	\$ff8a0a	WORD	Blitter
HALFTONE6	\$ff8a0c	WORD	Blitter
HALFTONE7	\$ff8a0e	WORD	Blitter
HALFTONE8	\$ff8a10	WORD	Blitter
HALFTONE9	\$ff8a12	WORD	Blitter
HALFTONE 10	\$ff8a14	WORD	Blitter
HALFTONE11	\$ff8a16	WORD	Blitter
HALFTONE12	\$ff8a18	WORD	Blitter
HALFTONE13	\$ff8a1a	WORD	Blitter
HALFTONE14	\$ff8a1c	WORD	Blitter
HALFTONE15	\$ff8a1e	WORD	Blitter
SCR_XINC	\$ff8a20	WORD	Blitter
SCR_YINC	\$ff8a22	WORD	Blitter
SRC_ADDR	\$ff8a24	LONG	Blitter
ENDMASK1	\$ff8a28	WORD	Blitter
ENDMASK2	\$ff8a2a	WORD	Blitter
ENDMASK3	\$ff8a2c	WORD	Blitter
DST_XINC	\$ff8a2e	WORD	Blitter
DST_YINC	\$ff8a30	WORD	Blitter
DST_ADDR	\$ff8a32	LONG	Blitter
X_COUNT	\$ff8a36	WORD	Blitter
Y_COUNT	\$ff8a38	WORD	Blitter
HOP	\$ff8a3a	BYTE	Blitter
OP	\$ff8a3b	BYTE	Blitter
GPIP	\$fffa01	BYTE	
AER	\$fffa03	BYTE	
DDR	\$fffa05	BYTE	
IERA	\$fffa07	BYTE	
IERB	\$fffa09	BYTE	
IPRA	\$fffa0b	BYTE	
	., .,		

IDDD	ducce o 1	DIVID	
IPRB	\$fffa0d	BYTE	
ISRA	\$fffa0f	BYTE	
ISRB	\$fffa11	BYTE	
IMRA	\$fffa13	BYTE	
IMRB	\$fffa15	BYTE	
VR	\$fffa17	BYTE	
TACR	\$fffa19	BYTE	
TBCR	\$fffa1b	BYTE	
TCDCR	\$fffa1d	BYTE	
TADR	\$fffa1f	BYTE	
TBDR	\$fffa21	BYTE	
TCDR	\$fffa23	BYTE	
TDDR	\$fffa25	BYTE	
SCR	\$fffa27	BYTE	
UCR	\$fffa29	BYTE	
RSR	\$fffa2b	BYTE	
TSR	\$fffa2d	BYTE	
UDR	\$fffa2f	BYTE	
KEYCTL	\$fffc00	BYTE	
KEYBD	\$fffc02	BYTE	
MIDICTL	\$fffc04	BYTE	
MIDI	\$fffc06	BYTE	
DISKCTL	\$ff8604	WORD	
FIFO	\$ff8606	WORD	
DMAHIGH	\$ff8609	BYTE	
DMAMID	\$ff860b	BYTE	
DMALOW	\$ff860b	BYTE	
GISELECT	\$ff8800	WORD	
GIREAD	\$ff8800	WORD	
GIWRITE	\$ff8802	WORD	
JOY FIRE	\$ff9200	WORD	STe
JOY POS	\$ff9202	WORD	STe
JOY0 X	\$ff9210	WORD	STe
JOY0_Y	\$ff9212	WORD	STe
JOY1_X	\$ff9214	WORD	STe
JOY1 Y	\$ff9216	WORD	STe
JOY2 X	\$ff9220	WORD	STe
JOY2 Y	\$ff9222	WORD	STe
DMA SOUND E	\$ff8900	WORD	STe
FBASEHI	\$ff8902	WORD	STe
FBASEMID	\$ff8904	WORD	STe
FBASELOW	\$ff8906	WORD	STe
CBASEHI	\$ff8908	WORD	STe
CBASEMID	\$ff890a	WORD	STe
CBASELOW	\$ff890c	WORD	STe
CDITOLLO VV	W110700	11 010	

EBASEHI	\$ff890e	WORD	STe
EBASEMID	\$ff8910	WORD	STe
EBASELOW	\$ff8912	WORD	STe
SOUND_CTRL	\$ff8920	WORD	STe
SOUND_DATA	\$ff8922	WORD	STe
SOUND_MASK	\$ff8924	WORD	STe

# Tests et affectations

Tests supportés:

== égal à

>= supérieur ou égal à

<= inférieur ou égal à</p>

> supérieur à

< inférieur à

!= différent de

Le résultat renvoyé par un test est -1 (vrai) ou 0 (faux).

#### ?: test?action si vrai:action si faux

syntaxe identique au langage C.

$$\phi$$
 d0==0?d1=0:d1=1 affecte 0 à d1 si d0 vaut 0, sinon affecte 1.

#### Affectations:

Les affectations de registre, mémoire ou variable (uniquement de type LA) sont possibles directement depuis l'évaluateur. La syntaxe est la suivante:

{adresse}=chaîne (affectation en mémoire)

{44e}=10

variable=chaîne

Tous les opérateurs arithmétiques sont doublés de l'opérateur d'affectation correspondant: +=, -=, \*=, /=, &=, |=, ^=, >>=,

<<=, ~=

syntaxe identique au langage C.

d0 += 1

ajoute 1 à d0 (équivaut à d0=d0+1).

# Fonctions prédéfinies

# mod(dividende, diviseur)

renvoie le reste de la division dividende/diviseur.

# sob(nomdevariable)

renvoie l'adresse de début du bloc désigné par nomdevariable.

#### eob(nomdevariable)

renvoie l'adresse de fin du bloc désigné par nomdevariable.

#### lob(nomdevariable)

renvoie la longueur du bloc désigné par nomdevariable.

# while(condition, action)

exécute action jusqu'à condition fausse, et renvoie faux(0). (peut être interrompu avec [Esc]).

# watch(adresse)

renvoie l'adresse calculée du branchement situé à adresse.

# 1prev(adresse)

renvoie l'adresse de la variable la plus proche de adresse.

# lnext(adresse)

renvoie l'adresse de la variable la plus proche de adresse.

# svar(nom)

crée une variable (de type LA).

# Format des données

L'évaluateur accepte les suffixes **.b** (octet), **.w** (mot) et **.1** (long mot) derrière toute constante, variable ou registre. Une limitation est à noter: Le **SR** et le **CCR** ne supportent aucun suffixe de taille. 

({a0}.w=1234)+(sr=a3.b)

Tout accès en mot ou long mot sur une adresse impaire est valide.

# Récursivité et limites de l'évaluateur

Les variables de type **LR** pouvant comporter des autoréférences, des références circulaires, ou encore trop de niveaux d'évaluations (plus de 50), un contrôle de la pile est effectué à chaque évaluation. En cas de dépassement, une erreur de syntaxe est indiquée.

# Priorité des opérateurs (en ordre décroissant)

<<
<< >>
2
\ ^ &
&
/
*
-
+
==
!=
>=
>= <=
<
>
=

# Appendice IV - Liste thématique des messages d'erreur

**b** signifie octet.

w signifie mot.

**1** signifie long mot.

s signifie chaîne.

d signifie décimal.

**x** signifie hexadécimal.

# Option non autorisée.

Utilisation dans la ligne de commande d'une option inexistante.

#### Plus de place en vblqueue.

Impossible d'installer une interruption en vblqueue.

# Erreur FATALE de réservation mémoire dans les préférences. Pressez une touche.

Une taille de tampon dans les préférences est trop grande pour la mémoire disponible.

## Erreur interne n°<bx> Décalage <lx>.

Une exception s'est produite dans Adebug. Elle peut être générée à la suite d'un écrasement partiel d'Adebug.

# Imprimante non prête.

Vérifiez les connections et l'état de l'imprimante.

#### Mémoire illisible.

L'adresse donnée n'est pas accessible en lecture.

# Attention! Copie incomplète.

Lors d'une copie, un ou plusieurs octets n'ont pu être copiés car l'adresse de destination est non inscriptible.

# Attention! Remplissage incomplet.

Lors d'un remplissage, un ou plusieurs octets n'ont pu être écrits car l'adresse de destination est non inscriptible.

# Adresse impaire ou illisible.

Dans une commande [Sft\_Alt\_0~9] l'adresse correspondant au registre concerné est impaire ou illisible.

# Pas de registres sauvés.

Une restauration de registres ([R]) est demandée sans qu'il y ait

eu au préalable de sauvegarde ([K]).

#### Recherche non définie.

Une recherche de la prochaine ([N]) ou de la dernière ([Alt\_N]) occurrence est demandée sans que la recherche ait été définie ([G]).

#### Mauvaise adresse de tampon.

Une adresse de tampon d'étiquettes illisible et/ou non inscriptible est fournie dans une commande [P].

#### Erreur disque.

Une erreur d'accès disque s'est produite.

## Opérations disque non autorisées en IPL>5.

Quand l'IPL interne d'Adebug est supérieur à 5, les accès au disque ne sont plus autorisés.

#### Chemin invalide.

Le chemin fourni est invalide (mal formulé et/ou inexistant).

#### Fichier <s> non trouvé.

Le nom de fichier fourni ne correspond à aucun fichier présent.

#### Erreur de lecture n°<d> dans le fichier <s>.

Une erreur d'accès disque en lecture s'est produite.

#### Erreur de création.

Impossible de créer le fichier demandé.

#### Erreur d'écriture n°<d> dans le fichier <s>.

Une erreur d'accès disque en écriture s'est produite.

#### Erreur d'écriture.

Une erreur d'accès disque en écriture s'est produite.

#### Pas un fichier exécutable.

Le nom donné dans la fonction **[Ct1\_L]** ne correspond pas à un fichier relogeable.

#### Erreur de relocation.

Le programme chargé avec la fonction **[Ct1\_L]** n'a pas une table de relocation correcte.

# Type de symboles inconnu.

Le programme chargé avec la fonction [Ctrl\_L] possède des

symboles de type inconnu (et qui ne seront donc pas chargés).

#### Plus de mémoire.

Il n'y a plus de mémoire disponible.

#### Mauvaise eval en Trace Jusqu'à.

L'expression donnée dans la fonction **[T]**race J)usqu'à comporte une évaluation incorrecte.

# Mauvaise eval en point d'arrêt.

L'expression attachée à un point d'arrêt comporte une évaluation incorrecte.

# Impossible de mettre le point d'arrêt.

L'adresse fournie est impaire et/ou illisible, ou il n'y a plus de point d'arrêt disponible.

#### Mauvaise pile pour le traçage.

La pile superviseur (SSP) est impaire, et/ou illisible, et/ou non inscriptible.

#### PC impair ou illisible.

Le PC ("Program Counter" ou compteur ordinal) est impair et/ou illisible.

#### Directive MAC inconnue.

Au cours d'une exécution de macro, une directive de macro inconnue est demandée.

# Tampon MAC plein.

Le tampon de macro est plein au cours d'un enregistrement ou d'un chargement.

#### Pas de MAC.

Pas de macro enregistrée ni chargée.

# `manquant dans macro.

Lors de l'exécution d'une macro, une fonction n'est pas précédée d'un '`' (backquote). Cela indique souvent une erreur de syntaxe AVANT l'endroit signalé.

#### Fonction inconnue.

En exécution de macro, une fonction (précédée par '`') n'existe pas dans Adebug.

#### Pas de VAR.

Pas de variable chargée ni créée.

#### Tampon VAR plein. Pressez une touche.

Le tampon contenant les références de variables est plein, à la suite du chargement ou de la création de variables.

#### Tampon LA plein. Pressez une touche.

Le tampon contenant les noms de variables est plein, à la suite du chargement ou de la création de variables.

#### Tampon LR plein. Pressez une touche.

Le tampon contenant l'expression des LR est plein, à la suite du chargement ou de la création de variables.

#### Tampon BL plein. Pressez une touche.

Le tampon contenant les BL est plein, ou le nombre maximum de BL est atteint, à la suite du chargement ou de la création de variables.

# Erreur n°<bd>,VAR <ld>,ligne <ld>:<s>.

Une erreur s'est produite lors de l'interprétation d'un fichier de variables (par exemple adebug.var).

Erreur 1: nom incorrect.

Erreur 2: type incorrect.

Erreur 3: évaluation incorrecte ou BL non trouvé.

#### Ro <s> non trouvée. Pressez une touche.

La routine <s> n'a pas été trouvée lors de son initialisation avec [Alt V].

#### Bl <s> non trouvé. Pressez une touche.

Le bloc <s> n'a pas été trouvé lors de son initialisation avec [Alt\_V].

# Appendice V - Petite documentation de l'Atari

Adr. Tai. Désignation Explication	Aur. Tai. Designation Explication
-----------------------------------	-----------------------------------

		C	ARTOUCHE
\$fa0000	12	cartridge_start	Adresse de base du port cartouche.
	8		
\$fbffff	Ko	cartridge_end	Adresse de fin du port cartouche.

			MEMOIRE
\$ff8001	В	memconf	Registre de configuration mémoire (bits 3-2 + 1-0)
			00: 128 Ko
			01: 512 Ko
			10: 2 Mo

		Δ	FFICHAGE
\$ff8201	В	dbaseh	Octet haut de l'adresse de la mémoire vidéo.
\$ff8203	В	dbasem	Octet médian de l'adresse de la mémoire vidéo.
\$ff820d	В	dbasel	Octet bas de l'adresse de la mémoire vidéo (STe /
ψΠΟΖΟΔ		ubasei	F030).
\$ff8205	В	vcounthi	Octet haut du compteur d'adresse vidéo (en
		{00}	écriture à partir du STe).
\$ff8207	В	vcountmid	Octet médian du compteur d'adresse vidéo (en
		{}	écriture à partir du STe).
\$ff8209	В	vcountlo	Octet bas du compteur d'adresse vidéo (en écriture
		{O}	à partir du STe).
\$ff820a	В	syncmode	
		{00}	
		0	0 = Synchro Interne, 1 = Externe.
		1	0 = Fréquence 50 Hz , 1 = 60 Hz.
\$ff820e	W	lineoffset	Offset jusqu'au début de ligne raster suivante
			(F030).
\$ff820f	В	linewidth	Largeur d'une ligne raster en words-1 (STE).
\$ff8210	W	linewidth	Largeur d'une ligne raster en words (F030).
\$ff8240	16	Palette de couleur.	
	W	{000}	Sur ST.
		{0000}	Sur STE et TT.
\$ff8260	В	shiftmode	Résolution vidéo (ST / STe):
		{00}	00 basse résolution 320*200 4 plans.
			01 moyenne résolution 640*200 2 plans.
****			10 haute résolution 640*400 1 plan.
\$ff8262	В	ttshiftmode	Résolution vidéo (TT):
		{ 000}	00 / 01 / 10: id ci-dessus
			100 résolution VGA 640*480 4 plans.
			110 haute résolution TT 1280*960 1 plans.
Φ.((O,O,O, 4	147	h. h. i.k 66	111 basse résolution TT 320*480 8 plans.
\$ff8264	W	hbitoff	Offset du bit de départ de l'adresse écran (F030).
\$ff8265	В	hbitoff	Offset du bit de départ de l'adresse écran (STe).
\$ff8266	W	spshift	Mode video (Bit 8: true color, Bit 4: plans) (F030).
\$ff8400	25	ttpalette	Palettes de couleur (256 fois 16 W) (TT) .
	6		

	AFFICHAGE F030				
\$ff8266	W	SPSHIFT	Bit 8: true color Bit 4: palette.		
\$ff8282	W	HHT	Nombre de demi-lignes.		
\$ff8284	W	HBB	Fin de ligne horizontale.		
\$ff8286	W	HBE	Début de ligne horizontale.		
\$ff8288	W	HDB	Overscan gauche horizontal.		
\$ff828a	W	HDE	Overscan droit horizontal.		
\$ff828c	W	HSS	Début de synchro horizontale.		
\$ff828e	W	HFS			
\$ff8290	W	HEE			
\$ff82a2	W	VFT	Nombre de champs.		
\$ff82a4	W	VBB	Fin d'écran.		
\$ff82a6	W	VBE	Début d'écran (en demi-lignes).		
\$ff82a8	W	VDB	Overscan haut.		
\$ff82aa	W	VDE	Overscan bas.		
\$ff82ac	W	VSS	Début de synchro verticale.		
\$ff82c0	W	RCO			
\$ff82c2	W	VCO	Bit 2: double pixel.		
			Bit 1: entrelacement.		
			Bit 0: double ligne.		

		DMA / FDC	C (WD 1772 - AJAX)
\$ff8604	B-L	diskctl	Registre de contrôle.
\$ff8606	B/W	fifo	Registre de données et de statut.
		{000000000}	
			Réservé.
		0	Etat ligne A0.
			Etat ligne A1.
		2	0 = FDC, 1 = HDC.
		3	0 = registre FDC, 1 = nb de secteurs.
		4	Reservé.
		5	0 = DMA actif.
		6	0 = HDC, 1 = FDC
		7	0 = lecture, 1 = écriture
		8	
\$ff8609	В	dmahigh	Octet haut de l'adresse du pointeur DMA.
\$ff860b	В	dmamid	Octet médian de l'adresse du pointeur DMA.
\$ff860d	В	dmalow	Octet bas de l'adresse du pointeur DMA.

		DMA TT SCSI
\$ff8701	В	Octet 3 adresse DMA.
\$ff8703	В	Octet 2 adresse DMA.
\$ff8705	В	Octet 1 adresse DMA.
\$ff8707	В	Octet 0 adresse DMA.
\$ff8709	В	Octet 3 compteur DMA.
\$ff870b	В	Octet 2 compteur DMA.
\$ff870d	В	Octet 1 compteur DMA.
\$ff870f	В	Octet 0 compteur DMA.
\$ff8710	V	Mot haut continue DMA.
\$ff8712	W	Mot bas continue DMA.
\$ff8714	W	Registre de contrôle DMA.

TT SCSI Contrôleur 5380			
\$ff8781	В	Contenu du bus de données SCSI.	
\$ff8783	В	Initialisation commande.	
\$ff8785	В	Début de transfert.	
\$ff8787	В	Commande cible.	
\$ff8789	В	Statut du bus.	
\$ff878b	В	Statut.	
\$ff878d	В	Donnée de commande du bus.	
\$ff878f	В	Reset des interruptions, erreur de parité, début init.	

	YM 2149			
\$ff8800	B-L	En lecture: giread En écriture: giselect	Lecture du port B du processeur sonore. Sélecteur de registre (\$e ou \$f) du processeur sonore.	
\$ff8802	B/W	En écriture: giwrite {00000000}                    0                1              3          5        5      6    7	Ecriture sur les ports A ou B du processeur sonore. Port A (registre \$e): Sélection de la face de disque. Sélection du disque A. Sélection du disque B. Signal RTS de la série. Signal DTR de la série. Signal STROBE de la parallèle. Sortie à usage général GPO (TT/F030: pas de HP interne). Réservé. Port B (registre \$f), les bits sont les lignes D0-D7	

	SON DMA (STE / TT / F030)				
\$ff8900	В	interrupt_ctrl	Contrôle des interruptions du buffer DMA.		
		{0000}			
			Interruption 15 du MPF (IPL7) en fin de buffer de		
		_   _ 0	replay.		
		1	Interruption 15 du MPF (IPL7) en fin de buffer de record.		
		- -	Interruption Timer A en fin de buffer de replay.		
		1 2	Interruption Timer A en fin de buffer de record.		
		3			
\$ff8901	В	dma_sound_enable	Registre de contrôle du son.		
		{0 00}	00 = arrête le son, 01 = joue le son une seule fois.		
			10 = reservé, 11 = joue le son en boucle.		
			0 = joue, 1 = enregistre.		
\$ff8903	В	fbasehi {oo}	Octet haut de l'adresse de base du son.		
\$ff8905	В	fbasemid {}	Octet médian de l'adresse de base du son.		
\$ff8907	В	fbaselo {o}	Octet bas de l'adresse de base du son.		
\$ff8909	В	cbasehi {oo}	Octet haut de l'adresse courante du son.		
\$ff890b	В	cbasemid {}	Octet médian de l'adresse courante du son.		
\$ff890d	В	cbaselo {o}	Octet bas de l'adresse courante du son.		
\$ff890f	В	ebasehi {oo}	Octet haut de l'adresse de fin du son.		
\$ff8911	В	ebasemid {}	Octet médian de l'adresse de fin du son.		
\$ff8913	В	ebaselo {o}	Octet bas de l'adresse de fin du son.		
\$ff8920	В	track_ctrl	Contrôle des pistes. (F030 seulement)		
			Bits 0-1 = numéro de piste (0-3) du DAC.		
255222			Bits 4-5 = nombre de pistes à jouer (1-4).		
\$ff8921	В	sound_ctrl	Mode et fréquence.		
			Bit 0-1 (fréquence):		
			00:6258 Hz.		
			01:12517 Hz.		
			10:25033 Hz.		
			11:50066 Hz.		
			Bit 7 (mode): 0 = stereo, 1 = mono.		
			Bit 6 (mode2): 0 = 8 bits, 1 = 16 bits (F030 seulement).		
\$ff8922	L	sound_data	Registre de donnée de l'interface Microwire.		
		{00000000}{000000000}	(Réglage du volume et de la tonalité).		
\$ff8924	L	sound_mask	Registre de masquage de l'interface Microwire.		
		{00000000}{00000000}	(voir table ci-dessous)		

Commande	Nom	Paramètre	Effet
011	main volume	000000	-80 dB
		010100	-40 dB
		101	0 dB
101	left volume	- 00000	-40 dB
		- 01010	-20 dB
		- 101	0 dB
100	right volume	- 00000	-40 dB
		- 01010	-20 dB
		-101	0 dB
010	treble	0000	-12 dB
		0110	0 dB
		1100	+12 dB
001	bass	0000	-12 dB
		0110	0 dB
		1100	+12 dB
000	mix	00	-12 dB
		01	Yamaha mix
		10	no mix
		11	reserved

	MATRICE (F030)				
\$ff8930	W	dev ctrl	Contrôle de la matrice.		
		Bit 0 (sortie DMA)	0 = handshake on, 1 =handshake off.		
		Bit 1-2	Horloge (voir ci-dessous).		
		Bit 3	0 = DMA in, 1 = tous les autres.		
		Bit 4 (sortie DSP)	0 = handshake on, 1 =handshake off.		
		Bit 5-6	Horloge (voir ci-dessous).		
		Bit 7	0 = déconnecté, 1 = connecté		
		Bit 8 (entrée externe)	0 = handshake on, 1 =handshake off.		
		Bit 9-10	Horloge (voir ci-dessous).		
		Bit 12 (entrée ADC)	0 = synchro interne, 1 = externe.		
\$ff8932	W	dest_ctrl	Contrôle de la matrice.		
		Bit 0 (entrée DMA)	0 = handshake on, 1 =handshake off.		
		Bit 1-2	Source (voir ci-dessous).		
		Bit 3	0 = DMA out, 1 = tous les autres.		
		Bit 4 (entrée DSP)	0 = handshake on, 1 =handshake off.		
		Bit 5-6	Source (voir ci-dessous).		
		Bit 7	0 = déconnecté, 1 = connecté		
		Bit 8 (sortie externe)	0 = handshake on, 1 =handshake off.		
		Bit 9-10	Source (voir ci-dessous).		
		Bit 12-13 (sortie DAC)	Source (voir ci-dessous).		
\$ff8934	В	ext_sync_div	Diviseur fréquence synchro externe.		
\$ff8935	В	int_sync_div	Diviseur fréquence synchro interne.		
\$ff8936	В	track_rec	Bits 0-1 = nombre de pistes à enregistrer (1-4).		
\$ff8937	В	adderin_input	Bit 0 = entrée ADC/DAC, Bit 1 = entrée		
#****************	_		multiplexeur.		
\$ff8938	В	channel_input	Bit 0 = droite, Bit 1 = gauche (0 = micro, 1 = PSG).		
\$ff8939	В	channel_amplification	Bits 0-3 = droite, Bits 4-7 = gauche.		
\$ff893a	В	channel_reduction	Bits 0-3 = droite, Bits 4-7 = gauche.		
\$ff8941	В	data_direction	Bits 0-2 (0 = in, 1 = out).		
\$ff8943	В	dev_data	Donnée		

HORLOGE		
00	25,175 MHz	
01	externe	
10	32 MHz	

SOURCE			
00	sortie DMA		
01	sortie DSP		
10	entrée externe		
11	entrée ADC		

HORLOGE TT				
\$ff8961 B clock_sel Sélection du registre.				
\$ff8963	В	clock_dat	Donnée du registre.	

	BLITTER					
\$ff8a00		blitbase	Adresse de base des registres du blitter.			
\$ff8a00 à \$ff8a1e	16W	halftone	Registres de demi-teinte 0 à 15. L'ensemble de ces 16 registres forme une trame de 16 points de large utilisée pour masquer les blocs graphiques gérés par le blitter.			
\$ff8a20	W	src_xinc	Nombre d'octets qui séparent les adresses de 2 mots à copier consécutifs d'une même ligne graphique.			
\$ff8a22	W	src_yinc	Nombre d'octets qui séparent 2 lignes consécutives du bloc graphique à copier dans une page écran.			
\$ff8a24	L	src_addr	Adresse du premier mot du bloc à copier.			
\$ff8a28	W	endmask1	Masque du premier mot d'une ligne du bloc à copier.			
\$ff8a2a	W	endmask2	Masque d'un mot du milieu (i.e. ni le premier mot, ni le dernier) d'une ligne du bloc à copier.			
\$ff8a2c	W	endmask3	Masque du dernier mot d'une ligne du bloc à copier.			
\$ff8a2e	W	dst_xinc	Même fonction que src_xinc (\$ff8a20) mais pour le bloc destination.			
\$ff8a30	W	dst_yinc	Même fonction que src_yinc (\$ff8a22) mais pour le bloc destination.			
\$ff8a32	L	dst_addr	Même fonction que src_addr (\$ff8a24) mais pour le bloc destination.			
\$ff8a36	W	x_count	Nombre de mots par ligne du bloc destination.			
\$ff8a38	W	y_count	Nombre de lignes du bloc destination.			
\$ff8a3a	В	НОР	Opération logique entre la trame de demi-teinte, définie par les registres de demi-teinte 0 à 15 (\$ff8a00 à \$ff8a1e), et le bloc source. 0: (source) OU (trame). 1: trame. 2: source. 3: (source) ET (trame).			

\$ff8a3b	В	OP {000-0000}	Opération logique entre le bloc source et le bloc de destination.  LINE NUMBER Définit le premier registre de demi-teinte utilisé (i.e. pour la première ligne du bloc).  SMUDGE Le positionnement de ce bit désactive le mode décrit ci-dessus. Le registre de demi-teinte utilisé est alors défini par les 4 bits de poids faibles de src_yinc.  HOG Drapeau permettant au blitter de disposer de tout ou moitié du temps d'occupation du bus.  BUSY Le positionnement de ce bit provoque l'exécution de la commande programmée. Ce bit est mis à 0
\$ff8a3d	В	{000000}	en fin d'exécution.  SKEW  Décalage, en bits, modulo 16 du bloc cible par rapport au bloc source.  NFSR  Positionné à 1, ce bit provoque la lecture d'un mot supplémentaire en début de ligne.  FXSR  Positionné à 0, ce bit provoque la lecture d'un mot supplémentaire en fin de ligne.

DMA SCC (MSTe/TT/F030)				
\$ff8c01	В	Octet 3 adresse DMA.		
\$ff8c03	В	Octet 2 adresse DMA.		
\$ff8c05	В	Octet 1 adresse DMA.		
\$ff8c07	В	Octet 0 adresse DMA.		
\$ff8c09	В	Octet 3 compteur DMA.		
\$ff8c0b	В	Octet 2 compteur DMA.		
\$ff8c0d	В	Octet 1 compteur DMA.		
\$ff8c0f	В	Octet 0 compteur DMA.		
\$ff8c10	V	Mot haut continue DMA.		
\$ff8c12	W	Mot bas continue DMA.		
\$ff8c14	W	Registre de contrôle DMA.		

SCC 8530 (MSTe/TT/F030)			
\$ff8c81	В	Registre de contrôle channel A.	
\$ff8c83	В	Registre de donnée channel A.	
\$ff8c85	В	Registre de contrôle channel B.	
\$ff8c87	В	Registre de donnée channel B.	

VME (MSTe/TT)			
Off0 = 0.4			,
\$ff8e01	В	sys_mask	Masque les états suivants:
		Bit 1	Interruption logicielle.
		Bit 2	HSYNC.
		Bit 4	VSYNC.
		Bit 5	SCC.
		Bit 6	MFP.
		Bit 7	_SYSFAIL sur le bus VME.
\$ff8e03	В	sys_stat	Statut des états précédents.
\$ff8e05	В	sys_int	Passer le bit 0 à 1 provoque une INT de niveau 1.
\$ff8e07	В	vme_mask	Masque les états suivants:
		Bit 1	IRQ1
		Bit 2	IRQ2
		Bit 3	IRQ3 (logicielle)
		Bit 4	IRQ4
		Bit 5	IRQ5 (SCC)
		Bit 6	IRQ6 (MFP)
		Bit 7	IRQ7
\$ff8e0d	В	vme_stat	Statut des états précédents.
\$ff8e0f	В	vme int	Passer le bit 0 à 1 provoque une INT de niveau 3.

CACHE MSTe					
\$ff8e21	\$ff8e21 B mste_cache Cache du méga STe.				

	JOYSTICK (STE)				
\$ff9201	В	joy_fire {0000}	Appui du bouton fire.		
\$ff9202	W	joy_pos	Positions: 3 1 2 0 {HBGDHBGDHBGDHBGDHBGDHBGD} (H:haut B:bas G:gauche D:droite)		
\$ff9211	В	joy0_x	Position x du joystick analogique 0.		
\$ff9213	В	joy0_y	Position y du joystick analogique 0.		
\$ff9215	В	joy1_x	Position x du joystick analogique 1.		
\$ff9217	В	joy1_y	Position y du joystick analogique 1.		
\$ff9221	В	joy2_x	Position x du stylo/pistolet optique.		
\$ff9223	В	joy2_y	Position y du stylo/pistolet optique.		

PALETTE (F030)			
\$ff9800	L	f030_palette	256 longs :
			Bits 2 à 7 : bleu.
			Bits 18 à 23 : vert.
			Bits 26 à 31 : rouge.

INTERFACE DSP HOST (F030)			
\$ffa200	В	host_icr	Interrupt Control Register.
\$ffa201	В	host_cvr	Command Vector Register.
\$ffa202	В	host_isr	Interrupt Status Register.
\$ffa203	В	host_ivr	Interrupt Vector Register.
\$ffa205	В	host_high	Octet haut du bus de transfert.
\$ffa206	В	host_mid	Octet médian du bus de transfert.
\$ffa207	В	host_low	Octet bas du bus de transfert.

			MFP
\$fffa00	В	mfp	Adresse de base du MFP 68901.
\$fffa01	В		Entrée/Sortie à Usage Général:
φιιιαυτ	Ь	gpip	Bit 0: BUSY du port parlièle.
			Bit 1: DCD (Data Carrier Detect) du port série.
			Bit 2: CTS (Clear To Send) du port série.
			Bit 3: blitter.
			Bit 4: clavier / midi.
			Bit 5: FDC / DMA.
			Bit 6: RING (sonnerie) du port série.
			Bit 7: moniteur monochrome.
\$fffa03	В	aer	Sélection du front (descendant : 0, montant : 1)
φιιιαυσ	В	acı	actif pour chaque bit de GPIP.
\$fffa05	В	ddr	Sens de circulation des données (entrée : 0, sortie:
φιιιαυσ	Ь	dui	1) pour chaque bit de GPIP.
\$fffa07	В	iera	Activation d'interruptions.
φιιιαυτ	Ь		Activation difficer aptions.
		{00000000}	Timer B.
		0	Erreur d'émission.
			Tampon d'émission vide.
		3	Erreur de réception.
			Tampon de réception plein.
			Timer A.
		6	Détection de sonnerie de la série.
		7	Détection du moniteur monochrome.
\$fffa09	В	ierb	Activation d'interruptions.
φιπαυσ	В	{00000000}	Activation difficultations.
			Signal BUSY de la parallèle.
			Signal DCD de la série.
		2	Signal CTS de la série.
		3	Blitter fini.
		4	Timer D.
		5	Timer C.
		6	ACIA Clavier et ACIA Midi.
		7	DMA.
\$fffa0b	В	ipra	Drapeau de mise en attente d'interruptions (id iera).
\$fffa0d	В	iprb	Drapeau de mise en attente d'interruptions (id ierb).
\$fffa0f	В	isra	Mise en service d'interruptions (id iera).
\$fffa11	В	isrb	Mise en service d'interruptions (id ierb).
\$fffa13	В	imra	Masque d'interruptions (id iera).
\$fffa15	В	imrb	Masque d'interruptions (id ierb).
\$fffa17	В	vr	macque a interruptions (ia ieib).
ψιιια ι Ι	D	{00000}	(Les bits 0 à 3 sont générés par le MFP).
		, ·	Drapeau d'interruption automatique.
		<u>                          </u>	Bit 4 du n° de vecteur d'interruption.
		5	Bit 5 du n° de vecteur d'interruption.
		6	Bit 6 du n° de vecteur d'interruption.
		7	Bit 7 du n° de vecteur d'interruption.
\$fffa19	В	tacr	Registre de contrôle du timer A.
\$fffa1b	В	tbcr	Registre de contrôle du timer A.
\$fffa1d	В	tcdcr	Registre de contrôle des timers C et D.
\$fffa1f	В	tadr	Registre de décompte du timer A.
\$fffa21	В	tbdr	Registre de décompte du timer A.
\$fffa23	В		Registre de décompte du timer B.  Registre de décompte du timer C.
		tcdr	
\$fffa25	В	tddr	Registre de décompte du timer D.
\$fffa27	В	ucr	Registre de contrôle de l'USART.

\$fffa2b	В	rsr	Registre d'état du récepteur.
		{00000000}                0                1          3        4        5      6     7	Validation du récepteur. Validation du caractère de synchronisation. Réception du caractère de synchronisation. "BREAK" détecté. Erreur de format. Erreur de parité. Surcharge. Tampon plein.
\$fffa2d	В	tsr {00000000}                 0                 3         4       5     6   7	Registre d'état du transmetteur.  Validation de l'émetteur.  Programmation de l'état de la sortie.  Emission d'un "BREAK".  Fin d'émission.  Validation automatique du récepteur.  Caractère à transmettre.  Tampon d'émission vide.
\$fffa2f	В	udr	Registre de données de l'USART.

	SFP004 (FPU)			
\$fffa40	W	FPCIR	Registre de statut.	
\$fffa42	W	FPCTL	Registre de contrôle.	
\$fffa44	W	FPSAVE	Registre de sauvegarde (inutilisé dans la librairie ALCYON).	
\$fffa46	W	FPREST	Registre de restauration (inutilisé dans la librairie ALCYON).	
\$fffa48	W	FPOPR	Registre de mot d'opération.	
\$fffa4a	W	FPCMD	Registre de commande.	
\$fffa4c	W	FPRES	Réservé.	
\$fffA4E	W	FPCCR	Registre de code condition.	
\$fffa50	L	FPOP	Registre d'opérande.	

			CLAVIER
\$fffc00	В	keyctl	
		En écriture: {00000000}                 }	Registre de contrôle de l'ACIA clavier.
		0	Vitesse de transmission.
		<u>                       </u>	Protocole série.
			Type de validation d'interruption.
		} 5	Drapeau interruptions permises.
		} 6    } 7	Registre d'état de l'ACIA clavier.
		En lecture:	Tampon plein.
		{00000000}	Donnée disponible.
		0	Signal DCD. Signal CTS.
		2	Erreur de transmission.
		3	Surcharge.
		4	Erreur de parité.
		5	Drapeau d'interruption.
		6    7	
\$fffc02	В	keybd	Registre de données de l'ACIA clavier.
\$fffc04	В	midictl	Registre d'état et de contrôle de l'ACIA midi. (Identique à celui de l'ACIA clavier (fffc00)).
\$fffc06	В	midi	Registre de données de l'ACIA midi.

	HORLOGE				
\$ffffc21	В	s_units	Unités des secondes.		
\$ffffc23	В	s_tens	Dizaines des secondes.		
\$ffffc25	В	m_units	Unités des minutes.		
\$ffffc27	В	m_tens	Dizaines des minutes.		
\$ffffc29	В	h_units	Unités des heures.		
\$ffffc2b	В	h_tens	Dizaines des heures.		
\$ffffc2d	В	weekday	Jour de la semaine.		
\$ffffc2f	В	day_units	Unités des jours.		
\$ffffc31	В	day_tens	Dizaines des jours.		
\$ffffc33	В	mon_units	Unités des mois.		
\$ffffc35	В	mon_tens	Dizaines des mois.		
\$ffffc37	В	yr_units	Unités des années.		
\$ffffc39	В	yr_tens	Dizaines des années.		
\$ffffc3b	В	cl_mod			
\$ffffc3d	В	cl_test			
\$ffffc3f	В	cl_reset			

	INFORMATIONS POST-MORTEM			
\$380	L	proc_lives	Drapeau de validité des informations post-mortem du système (\$12345678 si données valides).	
\$384	8L	proc_dregs	Espace de sauvegarde des registres d0 à d7 lors de l'effondrement du système.	
\$3a4	8L	proc_aregs	Espace de sauvegarde des registres A0 à A6 et SSP lors de l'effondrement du système.	
\$3c4	В	proc_enum	Numéro d'exception.	
\$3c8	L	proc_usp	Valeur de la pile utilisateur lors de l'effondrement.	
\$3cc	L	proc_stk	16 mots du dessus de la pile superviseur.	

			PAGE 4
\$400	L	etv_timer	Vecteur timer C de maintenance du multitâche du GEM.
\$404	L	etv_critic	Vecteur pointant sur une routine appelée en cas d'erreur disque.
\$408	L	etv_term	Vecteur pointant sur une routine de fin de processus appelée juste avant la routine TERM (Normalement pointe sur RTS).
\$40c	L	etv_xtra	Espace pour 5 vecteurs réservés pour extensions futures.
\$420	L	memvalid	Drapeau indiquant la validité des données mémoires de la page 4 (\$400\$500). Si sa valeur n'est pas \$752019F3 le prochain reset provoquera un démarrage à froid, donc avec entre autres un effacement complet de la mémoire.
\$424	В	memcntrl	Contient une copie du registre de configuration d'adresse memconf (\$ffff8001).
\$426	L	resvalid	Drapeau indiquant la validité de l'adresse de la routine reset contenue en \$42A. La routine est exécutée si resvalid contient \$31415926.
\$42a	L	resvector	Contient l'adresse de la routine reset exécutée si \$426 contient \$31415926.
\$42e	L	phystop	Fin de la RAM physique.
\$432	L	_membot	Début de la RAM utilisateur (TPA).
\$436	L	_memtop	Fin de la RAM utilisateur (TPA).
\$43a	L	memval2	Deuxième drapeau de validité de la mémoire (Cf \$420). Sa valeur normale est \$237698AA.
\$43e	В	flock	Drapeau qui bloquant la routine Vbl de désélection de disque quand sa valeur est non nulle.
\$440	W	seekrate 0:	Vitesse de déplacement piste à piste de la tête de lecture du drive utilisée lors des appels disque de la ROM. 6 ms.
		1: 2: 3:	12 ms. 2 ms (à ne pas utiliser sur ST). 3 ms (à ne pas utiliser à partir du STE).
\$442	W	timr ms	Délai (en ms) entre 2 appels de etv_timer (\$400).
\$444	W	_fverify	Drapeau de vérification des accès disques en écriture (pour RWABS (BIOS 4)).
\$446	W	_bootdev	Numéro du drive à partir duquel le système à été chargé et à partir duquel il sera chargé au prochain démarrage.
\$448	W	palmode	Drapeau de mode NTSC(60Hz) / PAL&SECAM (50Hz).
\$44a	W	defshiftmd	Résolution utilisée lors de la commutation de monochrome en couleur.

¢440	D	ashiffmd	Décolution graphique de l'application sourante II
\$44c	В	sshiftmd	Résolution graphique de l'application courante. Il s'agit de la copie du registre vidéo d'adresse
			\$ffff8260.
\$44e	L	_v_bas_ad	Adresse de la mémoire vidéo logique. Doit être
			multiple de 256 sur ST. Cette limitation n'a plus
			cours à partir du STE.
\$452	W	vblsem	Drapeau d'interdiction de la gestion Vbl du système
			(bloque en particulier les routines en liste Vbl).
\$454	W	nvbls	Nombre de vecteurs de la liste Vbl.
\$456	L	_vblqueue	Pointeur sur le premier vecteur de la liste Vbl.
\$45a	L	colorptr	Si non nul, pointeur sur la palette qui sera initialisée
			lors de la prochaine gestion de l'interruption Vbl par le système.
\$45e	L	screenpt	Contient l'adresse de la mémoire vidéo physique
			qui sera initialisée (si elle est non nulle) lors de la
			gestion Vbl suivante.
\$462	L	_vbclock	Compteur d'interruptions Vbl.
\$466	L	_frclock	Compteur d'interruptions Vbl traitées par le système.
\$46a	L	hdv init	Vecteur pointant sur une routine d'initialisation,
			déterminant les lecteurs existant sur l'ordinateur.
\$46e	L	swv vec	Vecteur pointant sur une routine exécutée lors de la
		_	connection ou de la déconnection du moniteur
			monochrome. Cette routine est appelée par la
			gestion Vbl du système. Contient par défaut
			l'adresse du reset en ROM. (i.e. {4}).
\$472	L	hdv_bpb	Pointeur sur une routine renvoyant l'adresse du
			bloc de paramètres du BIOS (ne concerne que le
			disque).Correspond à Get_BPB (BIOS 7).
\$476	L	hdv_rw	Vecteur pointant sur la routine de lecture/écriture
			sur disque. Correspond à RWABS (BIOS 4).
\$47a	L	hdv_boot	Vecteur pointant sur une routine chargeant le secteur de boot d'un disque.
\$47e	L	hdv_mediach	Vecteur pointant sur la routine gérant les
			changements de disque. Correspond à MEDIACH (BIOS 9).
\$482	W	_cmdload	Si ce drapeau est positionné lors du boot, le
			système tente de charger le programme COMMAND.PRG.
\$484	В	conterm	Drapeaux permettant de configurer le clavier:
		{0000}	Druit de la procesion divine touche (times C)
			Bruit de la pression d'une touche (timer C). Répétition des touches.
			Bruit par pression sur [Ctl_G].
			CONIN (BIOS 2) délivre l'état des touches Alt, Ctl,
		I	Sft et CapsLock dans les bits 24 à 31.
\$486	L	trp14ret	Registre temporaire de sauvegarde de l'adresse de
			retour lors d'un appel XBIOS.
\$48e	4L	themd	Pointeur sur le début d'une liste chainée qui devrait
			référencer les possibles TPAs dans des versions
			ultérieures de GEMDOS. Correspond à GET_MPB
			(BIOS 0).
\$49e	L	md	Sera utilisé quand la condition ci-dessus sera vérifiée.
\$4a2	L	savptr	Pointeur sur un espace de sauvegarde temporaire
¢4-0	14/	ndlaw -	des registres lors d'un appel BIOS (46 octets).
\$4a6	W	_nflops	Nombre de lecteurs de disquette connectés.
Φ4-C		222 24-4-	0: un lecteur ou aucun, 2: deux lecteurs.
\$4a8	L	con_state	Vecteur pointant normalement sur la routine
			d'affichage standard. Correspond par défaut à
			CONOUT (BIOS 3).

\$4ac	W	save_row	Sauvegarde de la ligne du curseur utilisé lors de l'appel de la fonction VT52 ESC Y (positionner curseur).
<b>C</b> 100	L	and and and	,
\$4ae	L	sav_context	Pointeur sur un espace de sauvegarde des registres utilisés lors d'un traitement d'exception.
\$4b2	L	_bufl	Pointeur sur une structure du GEMDOS utilisé pour la gestion des secteurs de données.
\$4b6	L		Pointeur sur la structure GEMDOS précédente utilisé pour la gestion des secteurs de la FAT et du Directory.
\$4ba	L	hz 200	Compteur 200 Hz incrémenté par le timer C.
\$4be	4B	the env	Chaine d'environnement par défaut, (4 octets nuls).
\$4c2	L	_drvbits	Table des lecteurs (disquettes, disques durs, ram- disques) gérés par le système.
\$4c6	L	_dskbufp	Pointeur sur le tampon disque dans lequel est chargé le secteur de boot et parfois utilisé par certaines routines graphiques (1 Ko).
\$4ca	L	_autopath	Pointeur sur une chaîne contenant le chemin d'accès utilisé au démarrage du système (AUTO ou NULL).
\$4ce	L	vbl list	Début de la liste Vbl (normalement 8 vecteurs).
\$4ee	W	_dumpflg	Sémaphore incrémenté par la gestion clavier du système lors de l'appui sur [Alt_Help]. Par défaut à -1. Cf scr_dump (\$502).
\$4f2	L	_sysbase	Pointeur sur le header du TOS.
\$4f6	L	_shell_p	Pointeur sur le Shell.
\$4fa	L	end_os	Pointeur sur la fin de la zone réservée au TOS en RAM (i.e. début de TPA d'un programme en AUTO).
\$4fe	L	exec_os	Pointeur sur le début de l'AES. Est utilisé après l'initialisation du BIOS.

			PAGE 5					
\$502	L	scr_dump	Pointeur sur une routine appelée si _dumpflg (\$4ee) est nul ou par la fonction HardCopy (XBIOS 20). Par défaut contient l'adresse de la routine de copie d'écran du système.					
\$506	L	prv_lsto	Pointeur sur une routine déterminant l'état du port parallèle. Utilisé par la routine de copie d'écran.					
\$50a	L	prv_lst	Pointeur sur une routine de sortie parallèle. Utilisé par la routine de copie d'écran.					
\$50e	L	prv_auxo	Pointeur sur la routine déterminant l'état du port série. Utilisé par la routine de copie d'écran.					
\$512	L	prv_aux	Pointeur sur la routine de sortie série. Utilisé par la routine de copie d'écran.					
\$516	L	pun_ptr	Pointe sur une structure si un disque dur est connecté et AHDI chargé.					
\$51a	L	memval3	Drapeau de validité de la mémoire (Cf 420).					
\$51e	8L	bconstat_vec	8 pointeurs sur la routine bconstat pour les divers périphériques (BIOS 1) (TOS>=1.2).					
\$53e	8L	bconin_vec	8 pointeurs sur la routine bconin pour les divers périphériques (BIOS 2) (TOS>=1.2).					
\$55e	8L	bcostat_vec	8 pointeurs sur la routine bcostat pour les divers périphériques (BIOS 8) (TOS>=1.2).					
\$57e	8L	bconout_vec	8 pointeurs sur la routine bconout pour les divers périphériques (BIOS 3) (TOS>=1.2).					
\$59e	В	long_frame	Drapeau 68030 (TOS>=1.6).					
\$5ac	L	prv_clk	Pointeur sur la routine produisant le bruit des touches du clavier (TOS>=1.6).					

LIGNE A											
Déc	Hexa										
-906	-38a	L	CUR FONT	Adresse du descripteur de la fonte courante.							
-856	-358	W	M_POS_HX	Position courante du point chaud de la souris en x.							
-854	-356	W	M_POS_HY	Position courante du point chaud de la souris en y.							
-852	-354	W	M_PLANES 1: -1:	Mode d'écriture du curseur de la souris: normal.  XOR.  Correspond à la fonction ligne A Transform Mouse (\$A00B).							
-850	-352	W	M_CDB_BG	Couleur des points du fond de la souris.							
-848	-350	W	M_CDB_FG	Couleur des points du premier plan de la souris.							
-846	-34e	32 W	MASK_FORM	Forme de la souris, 16 fois fond/premier plan.							
-782	-30e	45 W	INQ_TAB	Résultat de l'appel VDI vq_extend.							
-692	-2b4	45 W	DEV_TAB	Résultat de l'appel VDI vq_extend (max X/Y souris).							
-602	-25a	W	M_X	Position courante de la souris en x.							
-600	-258	W	M_Y	Position courante de la souris en y.							
-598	-256	W	M_HID_CT	Nombre d'appels actuels de Hide Mouse (\$A009).							
-596	-254	W	M_BUT Bit 0 : Bit 1 :	Etat courant du bouton de la souris. bouton gauche. bouton droit.							
-594	-252	48 W	REC_COL	Trois fois 16 mots correspondant à la répartition RVB des couleurs. Correspond à la fonction VDI vq_color.							
-498	-1f2	15 W	SIZ_TAB	Paramètres VDI pour les sorties de texte, de lignet de marqueur.							
-464	-1d0	L	CUR_WORK	Pointeur sur la structure d'attribut de la station de travail actuelle.							
-460	-1cc	L	DEF FONT	Pointeur sur le descripteur de fonte par défaut.							
-456	-1c8	4L	FONT_RING	Tableau de 4 pointeurs sur une liste chaînée de descripteurs de fontes.							
-440	-1b8	W	FONT_COUNT	Nombre de fontes dans la liste ci-dessus.							
-348	-15e	В	CUR_MS_STAT Bit 0: Bit 1: Bit 2~4: Bit 5: Bit 6: Bit 7:	Etat courant de la souris: bouton gauche. bouton droit. réservés. souris déplacée. modification de l'état du bouton droit. modification de l'état du bouton gauche.							
-346	-15a	W	V_HID_CNT	Nombre d'appels actuels de Hide Cursor.							
-344	-158	W	CUR_X	Coordonnée x du prochain dessin de la souris.							
-342	-156	W	CUR_Y	Coordonnée y du prochain dessin de la souris.							
-340	-154	W	CUR_FLAG	Si non nul, la souris sera redessinée à la prochaine Vbl du système.							
-339	-153	В	MOUSE_FLAG	Si non nul, la souris est gérée par la Vbl du système.							
-334	-14e	W	V_SAV_X	Sauvegarde de la coordonnée x du curseur.							
-332	-14c	W	V_SAV_Y	Sauvegarde de la coordonnée y du curseur.							
-330	-14a	W	SAVE_LEN	Sauvegarde de la hauteur de la forme de la souris.							
-328	-146	L	SAVE_ADR	Adresse de sauvegarde de la forme de la souris.							
-324	-144	W	SAVE_STAT Bit 0: Bit 1:	Etat de la sauvegarde des informations souris.  Validité du contenu du tampon de sauvegarde.  Largeur des données sauvegardées (0:Mot/1:Long mot).							

200	4.40	400	CAVE ADEA	Convenende du ford la 1 (40 )
-322	-142	128 W	SAVE_AREA	Sauvegarde du fond sous la souris (16 mots par plan,*4 plans).
-66	-42	L	SYSTEM_TIM	Pointeur sur une routine appelée à chaque interruption Timer C du système.
-62	-3e	L	USER_TIM	Pointeur supplémentaire identique à SYSTEM_TIM.
-58	-3a	L	USER_BUT	Pointeur sur une routine appelée par la Vbl du système en cas de click souris.
-54	-36	L	USER_CUR	Pointeur sur une routine appelée par la Vbl du système pour la gestion de l'affichage de la souris.
-50	-32	L	USER_MOT	Pointeur sur une routine appelée par la Vbl du système pour la gestion du déplacement de la souris.
-46	-2e	W	PIXEL_HEIGHT	Hauteur d'un caractère dans la résolution courante.
-44	-2c	W	MAX_CELL_X	Nombre de caractères par ligne -1.
-42	-2a	W	MAX CELL Y	Nombre de lignes de texte -1.
-40	-28	W	NEXT_CELL_OFFS ET	Nombre d'octets par ligne de texte (BYTES_LN * PIXEL_HEIGHT ).
-38	-26	W	BG COLOUR	Couleur de fond du texte.
-36	-24	W	FG COLOUR	Couleur d'encre du texte.
-34	-22	L	CURSOR ADDR	Adresse du curseur de texte.
-30	-1e	W	FIRST_CELL_OFF SET	Nombre d'octets du début de l'écran à la première ligne de texte.
-28	-1c	W	CURSOR X	Coordonnée x du curseur de texte.
-26	-1a	W	CURSOR Y	Coordonnée y du curseur de texte.
-24	-18	В	CURSOR_FLASH	Fréquence de clignotement du curseur de texte (en nombre de VBL).
-23	-17	В	CURSOR_TIMER	Compteur interne de clignotement du curseur de texte.
-22	-16	L	MONO_FONT	Adresse de la fonte courante.
-18	-12	W	LAST_ASCII	Code ASCII du dernier caractère de la police courante.
-16	-10	W	FIRST_ASCII	Code ASCII du premier caractère de la police courante.
-14	-е	W	FONT_WIDTH	Largeur d'un caractère de la police courante.
-12	-C	W	MAX_X	Largeur de l'écran en pixel.
-10	-a	L	FONT_ADDR	Pointeur sur un tableau contenant les adresses des trois polices du système.
-6	-6	В	ALPHA_STATUS	Drapeaux du mode d'affichage: bit 0: curseur clignotant (blink) bit 1: curseur affiché (display) bit 2: curseur actif (on) bit 3: césure en fin de ligne (line wrap) bit 4: texte inversé (inverse video) bit 5: position curseur sauvée (avec Esc J) bit 6: après clignotement (blinked)
-4	-4	W	MAX_Y	Hauteur de l'écran en pixel.
-2	-2	W	BYTES_LN	Nombre d'octets par ligne.
+0	+0	W	VPLANES	Nombre de plans vidéo.
+2	+2	W	VWRAP	Nombre d'octets par ligne écran.
+4	+4	L	CONTRL	Adresse du tableau CONTRL.
+8	+8	ī	INTIN	Adresse du tableau INTIN.
+12	+c	ī	PTSIN	Adresse du tableau PTSIN.
+16	+10	ī	INTOUT	Adresse du tableau INTOUT.
<u> </u>	<del> </del>	+		

+0	+0	W	VPLANES	Nombre de plans vidéo.							
+2	+2	W	VWRAP	Nombre d'octets par ligne écran.							
+4	+4	L	CONTRL	Adresse du tableau CONTRL.							
+8	+8	L	INTIN	Adresse du tableau INTIN.							
+12	+c	L	PTSIN	Adresse du tableau PTSIN.							
+16	+10	L	INTOUT	Adresse du tableau INTOUT.							
+20	+14	L	PTSOUT	Adresse du tableau PTSOUT.							
+24	+18	W	COLBIT0	Valeur du plan 0 de la couleur.							
+26	+1a	W	COLBIT1	Valeur du plan 1 de la couleur.							

<b>TJ0</b>	<b>41</b> 0	W	COLDITA	Valour du plan 2 de la soulour
+28	+1c +1e	W	COLBIT2 COLBIT3	Valeur du plan 2 de la couleur.  Valeur de plan 3 de la couleur.
+30		W		
+34	+20 +22	W	LSTLIN LNMASK	Vaut -1 (\$FFFF).
+36	+24	W	WMODE	Style de ligne.  Mode d'écriture (0 à 3).
+38		W		
	+26		X1   Y1	Abscisse du point n°1.
+40	+28	W		Ordonnée du point n°1.
+42	+2a	W	X2 Y2	Abscisse du point n°2.
+44	+2c	L		Ordonnée du point n°2.
+46	+2e		PATPTR	Adresse du motif de remplissage.
+50	+32	W	PATMSK	Masque de remplissage.
+52	+34		MFILL	Drapeau de remplissage monochrome (0: mono 1: couleur).
+54	+36	W	CLIP	Drapeau de clipping (0: pas de clipping).
+56	+38	W	XMINCL	Abscisse du coin supérieur gauche du rectangle de clipping.
+58	+3a	W	YMINCL	Ordonnée du coin supérieur gauche du rectangle de clipping.
+60	+3c	W	XMAXCL	Abscisse du coin inférieur droit du rectangle de clipping.
+62	+3e	W	YMAXCL	Ordonnée du coin inférieur droit du rectangle de clipping.
+64	+40	W	XDDA	Coefficient d'arrondi en principe \$8000.
+66	+42	W	DDAINC	Coefficient de déformation (multiplié par \$100).
+68	+44	W	SCALDIR	Drapeau de déformation (0: rétrécissement, 1: agrandissement).
+70	+46	W	MONO	Drapeau de proportionnalité de la fonte. (0: non proportionnelle).
+72	+48	W	SRCX	Abscisse du caractère courant dans la fonte.
+74	+4a	W	SRCY	Inutilisé.
+76	+4c	W	DSTX	Abscisse du caractère courant sur l'écran.
+78	+4e	W	DSTY	Ordonnée du caractère courant sur l'écran.
+80	+50	W	DELX	Largeur d'un caractère.
+82	+52	W	DELY	Hauteur d'un caractère.
+84	+54	L	FBASE	Adresse de la fonte de caractères (à ne pas utiliser sous GDOS ni TURBO ST).
+88	+58	W	FWIDTH	Largeur de la fonte.
+90	+5a	W	STYLE	Drapeaux d'effets spéciaux:
			{00000}	Gras Grisé Italique Souligné Surligné
+92	+5c	W	LITEMSK	Masque de grisé.
+94	+5e	W	SKEWMSK	Masque d'italique.
+96	+60	W	WEIGHT	Epaisseur des caractères gras.
+98	+62	W	ROFF	Décalage supérieur pour l'italique.
+100	+64	W	LOFF	Décalage inférieur pour l'italique.
+102	+66	W	SCALE	Echelle. (0: normal).
+104	+68	W	CHUP	Angle de rotation en dixièmes de degrés.
+106	+6a	W	TEXTFG	Couleur du texte.
+108	+6c	L	SCRTCHP	Adresse d'un tampon pour les effets spéciaux.
+112	+72	W	SCRPT2	Offset du tampon d'agrandissement dans SCRTCHP.
+114	+74	W	TEXTBG	Couleur de fond pour le texte.
+116	+76	W	COPYTRAN	Drapeau pour la fonction de copie de rasters (0: opaque sinon transparent).

+118	+78	L	SEEDABORT	Pointeur sur la routine de test d'arrêt du remplissage.
+126	+7e	L	SETSCR	Pointeur sur une routine appelée par Setscreen()

# **Appendice VI - Liste des figures**

Figure 1: les cinq fenêtres d'Adebug en cinq types de visualisation.

Figure 2: une fenêtre en mode désasemblage plein écran ([Alt\_Z]).

Figure 3: le mode quatre fenêtres par élimination de la fenêtre 1([Alt\_S]).

Figure 4: le mode visualisation ASCII du contenu des registres ([Ctl\_0~7]).

Figure 5: le mode édition du SR et des timers ([Alt\_E]).

Figure 6: le mode assemblage en ligne ([Alt\_E]).

Figure 7: le mode édition de mémoire en hexadécimal ([Alt\_E]).

Figure 8: le mode édition de mémoire en ASCII ([Alt\_E]).

Figure 9: liste des tailles courante et maximale des tampons de variables ([Help]).

Figure 10: liste des variables ([L]).

Figure 11: visualisation d'une stack frame ([Ctl\_Alt\_F]).

Figure 12: liste des points d'arrêt ([Ctl\_Alt\_B]).

Figure 13: historique des instructions ([H]).

Figure 14: exceptions du 68xxx.

Figure 15: vecteurs utilisateurs sur Atari.

Figure 16: codage de la vitesse du port série.

Figure 17: codage de la parité du port série.

# Appendice VII - Table ASCII

Φ	0	1	2	3	4	5	6	7	8	9	A	В	C	D	Ε	F
0		0		0	0	Р	`	р	Ç	É	á	ã	ij	0	α	<b>=</b>
1	슌	:	į.	1	A	Q	а	q	ü	æ	í	õ	IJ	Ш	β	±
2	ς>	2	ш	2	В	R	b	۲	é	Æ	ó	Ø	X	9	Γ	2
3	٥	3	#	3	C	S	С	s	â	ô	ú	.0	1	Z	π	≤
4	¢	¥	\$	4	D	T	d	t	ä	ö	ñ	œ	7	7	Σ	ſ
5	Ø	5	Z	5	Ε	U	е	=	à	ò	Ñ	Œ	T	٦	σ	J
6	Z	8	&	6	F	Ų	f	>	å	û	<u>a</u>	À	п	U	Д	÷
7	0	7	•	7	G	M	g	×	Ç	ù	ō	Ã	1	Л	τ	n
8	<b>V</b>	8	(	8	Н	X	h	X	ê	ÿ	ż	õ	7	1	Φ	0
9	0	9	)	9	Ι	Y	i	y	ë	Ö	-		П	٦	Θ	•
A	ŧ	а	*	:	J	Z	j	z	è	Ü	7	1	נו	1	Ω	•
В	4	E S	+	;	К	[	k	{	ï	¢	纟	Ť	•	٩	δ	<b>√</b>
C	F		,	<	L	V	1	_	î	£	丬	q	כ	٩	Ф	n
D	C/R	ב	-	=	M	1	M	}	ì	¥	i	©	ל	§	ф	2
Ε	$\overline{}$	×		>	N	٨	n	2	Ä	β	«	ß	n	*	ε	3
F	K	ø	7	?	0	_	0	Δ	Å	f	>>	тм	J	00	Π	-

# Appendice VIII - Index général

```
^ 107
~ 107
!= 112
& 107
&= 113
(107
107
* 106
*= 113
+ 106
++ 106
+= 113
- 106
-- 106
-= 113
/ 106
/=113
1 Introduction 7
1.1 Contenu de la disquette 7
1.2 Qu'est-ce qu'un débogueur 8
1.3 Qu'est-ce qu'un débogueur symbolique 9
1.4 Connaissances nécessaires 9
2 - Disque (souple et dur) 103
2 Description rapide 10
2.1 Introduction 10
2.2 Adebug, un programme n'utilisant pas GEM 12
2.3 Saisie d'une commande - Ligne de commande 14
2.4 Variables 15
2.5 Macros 16
2.6 Configuration 17
2.7 Ligne de commande à l'exécution 17
2.8 Usage d'un terminal 18
2.9 Exemple d'utilisation 19
3 - Le port série (RS232) 103
3 Références 22
3.1 Liste thématique des fonctions 22
3.10 Opérations de contrôle de flux du programme 67
3.11 Points d'arrêt 73
3.12 Commandes diverses 78
3.13 Profiler (échantillonneur de temps d'exécution) 82
3.2 Liste alphabétique des fonctions 26
3.3 Fenêtres 29
3.4 Gestion et édition mémoire 37
3.5 Macros 43
```

```
3.6 Variables 48
3.7 Écran 57
3.8 Disque 58
3.9 Configuration 61
4 - Le port parallèle (imprimante) 105
< 112
<< 107
<<= 113
<= 112
^= 113
~= 113
== 112
> 112
>= 112
>> 107
>>= 113
? 112
A0 à A7 109
A0~A6 31
A0~A7 31; 32
adebug.sav 17; 61
adebug.var 9; 48; 52
adressage absolu court 91
adressage absolu long 91
adressage direct d'un registre d'adresse 88
adressage direct d'un registre de donnée 88
adressage en double indirection d'un registre avec index et
déplacement sur 32 bits 90
adressage en double indirection du PC avec index et déplacement
sur 32 bits 92
adressage immédiat 88
adressage indirect d'un registre avec index et déplacement sur 32
bits 90
adressage indirect d'un registre d'adresse 89
adressage indirect d'un registre d'adresse avec déplacement 89
adressage indirect d'un registre d'adresse avec index et
déplacement 90
adressage
            indirect
                      d'un
                              registre
                                        d'adresse
                                                    avec
                                                           post-
incrémentation 89
adressage
            indirect
                       d'un
                              registre
                                         d'adresse
                                                            pré-
                                                     avec
décrémentation 89
adressage indirect du PC avec index et déplacement sur 32 bits
91
adressage relatif au PC avec déplacement 91
adressage relatif au PC avec index et déplacement 91
adresse 84
Adresse impaire 101
```

Adresse impaire ou illisible 32; 33; 115

adresser 84

**AES 76** 

Alternate 10

alt\_ 16

app 58

Appendice I - Connaissances nécessaires à l'usage d'un

débogueur 83

Appendice II - Périphériques utilisables 103

Appendice III - L'évaluateur 106

Appendice IV - Liste thématique des messages d'erreur 115

Appendice V - Petite documentation de l'Atari 119

Appendice VI - Liste des figures 136

Appendice VII - Table ASCII 137

**APPENDICES 83** 

architecture 85

ASCII 12; 31

assembleur 9

Attention! Copie incomplète 41; 115

Attention! Remplissage incomplet 41; 115

Avant-propos 2

b 114

BL 49; 61

Bl <s> non trouvé 118

bloc 59

bus 84

bus d'adresse 84

bus de donnée 84

C 37

C)harger 46

**CAAR 109** 

**CACR 109** 

CCR 69; 85; 109; 114

Chemin invalide 116

commentaire 46

compteur 82

Control 10

**CRP 109** 

ctl\_16

cw 107

D0 à D7 109

désassemblage 12; 31

destination 88

**DFC 109** 

Directive MAC inconnue 117

donnée 84

drapeaux 103

"Dump" ASCII 31 "Dump" hexadécimal 31 dump' hexadécimal 12 E)nregistrement 43 E)nregistrer 44 E)nregistrer T)racer A)rrière P)asser C)lr 44 Ecran 103 entrées/sorties 93 eob 113 EQ 48 Erreur d'écriture 116 Erreur d'écriture n°<d> dans le fichier <s>. 116 Erreur de bus 101 Erreur de création 116 Erreur de lecture n°<d> dans le fichier <s>. 116 Erreur de relocation 116 Erreur disque 116 Erreur FATALE de réservation mémoire dans les préférences 115 Erreur interne n°<bx> Décalage <lx>. 115 Erreur n°<bd>,VAR <ld>,ligne <ld> <s>. 118 état 98 évaluateur 106 EX 51 exceptions 9; 75 faux 112 fenêtre active 12; 29 fenêtre courante 12; 29; 70; 80 fenêtres 29: 107 Fichier <s> non trouvé 116 fichier relogeable 58 flèche 95 Fonction inconnue 118 Fonctions prédéfinies 113 Format des données 114 Formats de constantes acceptés 106 FP0 à FP7 109 FP0~FP7 32 **FPCR 109 FPIAR 109 FPSR 109** gtp 58 hexadécimal 31 HIS 15; 61 historique 15 Impossible de mettre le point d'arrêt 73; 117 Impression système 105

imprimante 80

Imprimante non prête 115

Imprimer 80

Indirections et parenthèses 107

Inspect 31

inspect (en mode source) 12

instruction 84

instructions d'entrées/sorties 94

instructions de branchement 94

instructions de manipulation d'1 bit 93

instructions de manipulation de bits 94

instructions de test 94

instructions de traitement arithmétique 93

instructions de traitement logique 93

instructions diverses 94

interruption 78; 97

interruptions 87

**IPL 69** 

**ISP 109** 

J)ouer 44; 45

J)ouer T)racer A)rrière P)asser V)oir D)ébut 44

1114

l'instruction illégale 96

LA 48; 61

LIFO 86

ligne de commande 14; 17

lnext 113

lob 113

logiques 103

lprev 113

LR 49; 61

MAC 61

macro 43

macros 16

marque 80

Mauvaise adresse de tampon 116

Mauvaise eval en point d'arrêt 117

Mauvaise eval en Trace Jusqu'à 117

Mauvaise pile pour le traçage 117

MC68xxx 97

mémoire 83

Mémoire illisible 115

mémoire morte 83

mémoire vive 83

microprocesseur 83

**MMUSR 109** 

mod 113

mode trace 9

modes d'adressage 88

MSP 109

multifenêtrage 12

N 37

opérande 88

Opérateurs arithmétiques 106

Opérateurs logiques 107

Opérations disque non autorisées en IPL>5 116

Option non autorisée 115

parité du port série 81

Pas de MAC 117

Pas de registres sauvés 115

Pas de VAR 118

Pas un fichier exécutable 116

Passages de paramètres dans les routines 108

PC 19; 32; 33; 85; 95; 109

PC impair ou illisible 117

périphérique 87

périphériques 9

physiques 103

pile 86

Plus de mémoire 117

Plus de place en vblqueue 115

point d'arrêt 74; 96

Point d'arrêt n°x atteint 96

Point d'arrêt n°x mis 96

points d'arrêt 9; 75; 107

port parallèle 105

préférences 73; 105

prg 58

Priorité des opérateurs 114

Program Counter 19

programme 85

programme exécutable 8; 58

raccourcis clavier 12

Recherche non définie 116

Récursivité et limites de l'évaluateur 114

registre 84

registres 84

registres d'adresse 84

registres de donnée 84

registres du 68xxx 12; 31

répertoire 58

RO 15; 49

Ro <s> non trouvée 118

rts 70; 71

S 37 séquentiel 85 SFC 109 sft\_ 16 Shift 10 sob 113 Source 31 source (en mode source) 12 sous-programme 98 sous-programmes. 86 SP 109 SR 37; 95; 109; 114 SRP 109 SSP 32; 33; 109 Status Register 95 STO 61 superviseur 87 svar 113 symboles 8 symbolique 16 système d'exploitation 85 T 37; 95 Tampon BL plein 118 Tampon LA plein 118 Tampon LR plein 118 Tampon MAC plein 117 Tampon VAR plein 118 TC 109 Tcsh 15 Temps réel 104 terminal 18; 81; 103 Tests et affectations 112 tests et branchements 93 timer 87 timers 37 tos 58 Trace 95; 101 tracer 95 traitement numérique 93 trap 2 traps 96 TTO 109 TT1 109 ttp 58

145

Type de symboles inconnu 116

type de visualisation 31

**USP 109** 

```
utilisateur 87
V 37
VAR 61
variables 9; 15; 31
variables (en mode source) 12
Variables, blocs, routines (LA, LR, EQ, EX, RO, BL) 108
VBR 109
VDI 76
vecteur 73; 98
videotexte 81
visualisation ASCII 60
vitesse 81
vrai 112
w 114
w1~w5 107
watch 113
while 113
X 37
Z37
[]
[0~7] Changer de type la fenêtre courante 31
[Alt_1~5] 31
[Alt_@] 81
[Alt A]SCII 60
[Alt_B]reakpoint 74
[Alt_D]irectory 58
[Alt_eX]écute 70
[Alt_E]dition de fenêtre 37
[Alt_Help]Impression d'écran 78
[Alt_L] 34
[Alt_M]emory free 78
[Alt_N] 42
[Alt_P]rint 80
[Alt_R]emettre à zéro le profiler 82
[Alt_S]épare/Regroupe fenêtres 30
[Alt_T]ype de visualisation 31
[Alt_V]ariable 48
[Alt_Z]oom de fenêtre 29
[A]dresse de fenêtre 34
[B]inary load 59
[Ctl_0~7] 31
[Ctl_8] 32
[Ctl 9] 32
[Ctl_Alt_1~5] 32
[Ctl_Alt_B] 77
[Ctl Alt D] 76
[Ctl_Alt_F] 72
```

[Ctl Alt G] 36

[Ctl\_Alt\_I]nverse 57

[Ctl\_Alt\_J]sr to subroutine 70

[Ctl Alt L] 36

[Ctl\_Alt\_R] 81

[Ctl\_Alt\_S] 36

[Ctl\_Alt\_V] Charger les symboles 53

[Ctl\_Alt\_Z] Trace T1 71

[Ctl\_A]rrêt 68

[Ctl\_B]reakpoint 73

[Ctl\_C] 67

[Ctl\_D]étourne 76

[Ctl\_eX]écute 70

[Ctl\_E]xception 75

[Ctl\_Flèche bas] 34

[Ctl\_Flèche droite] 34

[Ctl\_Flèche gauche] 33; 34

[Ctl\_Flèche haut] 34

[Ctl\_F]orce 70

[Ctl Home] 36

[Ctl\_I] 36

[Ctl J]ump 70

[Ctl\_K]ill 75

[Ctl\_L]oad program 58

[Ctl\_P]références 62

[Ctl\_R]un 68

[Ctl\_S]kip 67

[Ctl\_T]race 70

Ctl\_Until 70

[Ctl\_V] 53

[Ctl\_Z] 67

[C]opier 41

[D]irectory 58

[Esc] 42

Elvalue 80

[F1] 80

[F2] 80

[F5] 11; 80

[Flèche bas] 33

[Flèche droite] 33

[Flèche gauche] 33

[Flèche haut] 33

[F]ill 41

[G]et expression 42

[Help] 54

[H]istorique 79

[I]nterrupt Priority Level 78

147

```
[J]ump 70
[K]eep 78
[L]iste des variables 55
[M]acro 44
[N]ext 42
[O]utput 81
[P]rint 80
[R]estore 78
[Sft_Alt_D]irectory 58
[Sft_Alt_Help] 71
[Sft Ctl 0~7] 32
[Sft_Ctl_8] 33
[Sft_Ctl_9] 33
[Sft_Ctl_Alt_J]sr to subroutine 71
[Sft_Ctl_C] 67
[Sft Ctl Home] 36
[Sft_Flèche bas] 34
[Sft Flèche haut] 34
[Sft_Watch] 35
[S]ave binary 60
[Tab] 29
[T]race (Jusqu'à, Instruction, Rien, 68020) 68
[U]ntil 69
[V]oir 57
[W]atch 35
 43
 manquant dans macro 117
{ 108
| 107
| = 113
} 108
° 46
°b 47
°c 46
°F 46
°1 46
°s 46
°t 46
```