

ATARI ST

A-DÉBOG

Le premier débogueur
symbolique professionnel



arobace éditions

ADEBOG

Mode d'emploi

Auteurs :

Alexandre Lemaesquier, Raphaël Lemoine,
Laurent Chemla

AROBACE Editions © 1990

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.

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 **Adébog**.

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'Adébog.

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

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

Adébog a été conçu avec le plus grand soin, et testé par de nombreuses personnes durant plusieurs mois. 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 plus particulièrement

Milan Bedov, Gilles Boriloux, Emmanuel Collignon, François Corroyer, Hervé Dudognon, Patrice Felix, Daniel Fustis, Kevin Godschalk, François "Guinness" Guillemé, Stéphane Hosemans, François Maine, Rodolphe Nadal, Sophie Netter, Jean-Marie Patry, Mathieu Picovschi, Alex Sancho, Pascal Truong.

Sommaire

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.....	10
1.4 Connaissances nécessaires	10
2 Description rapide.....	11
2.1 Introduction	11
2.2 Adébog, 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	18
2.7 Ligne de commande à l'exécution.....	18
2.8 Usages d'un terminal.....	19
2.9 Version cartouche.....	20
2.10 Changement de moniteur en cours de session	20
2.11 Exemple d'utilisation	21
3 Références.....	24
3.1 Liste thématique des fonctions.....	24
3.2 Liste alphabétique des fonctions.....	28
3.3 Fenêtres	32
[Alt_Z]oom de fenêtre.....	32
[Tab].....	33
[Alt_S]épare/Regroupe fenêtres.....	33
[Alt_T]ype de visualisation	34
[Alt_1~5].....	34
[Ctl_0~7].....	35
[Ctl_8].....	36
[Ctl_9].....	36
[Sft_Ctl_0~7].....	36
[Sft_Ctl_8].....	36
[Sft_Ctl_9].....	36
[Flèche haut].....	36

[Flèche bas]	36
[Flèche gauche]	37
[Flèche droite]	37
[Sft flèche haut]	37
[Sft flèche bas]	37
[Ctl flèche haut]	37
[Ctl flèche bas]	37
[A]dresse de fenêtre	37
[Alt_L]ock	37
3.4 Gestion et édition mémoire	39
[Alt_E]dition de fenêtre	39
[C]opier	41
[F]ill	41
[G]et expression	42
[N]ext	42
[Alt_N]	42
3.5 Macros	43
[M]acro	44
3.6 Variables	48
[Alt_V]ariable	48
[Help]	55
[L]iste des variables	57
3.7 Ecran	59
[V]oir	59
[Ctl_Alt_I]nverse	59
[Ctl_O]utput	59
[F8]	59
[F9]	59
[F10]	59
3.8 Opérations disque	60
[D]irectory	60
[Alt_D]irectory	60
[Ctl_L]oad program	60
[B]inary load	61
[Alt_A]SCII	62
[S]ave binary	62

3.9 Configurations.....	63
[Ctl_P]références.....	64
3.10 Opérations de contrôle de flux du programme	
.....	67
[Ctl_C].....	67
[Ctl_Z].....	67
[Ctl_S]kip.....	67
[Ctl_R]un.....	67
[Ctl_A]rrêt.....	68
[T]race (Jusqu'à, Instruction, Rien, 68020).....	68
[U]ntil.....	70
[Ctl_U]ntil.....	70
[J]ump.....	70
[Ctl_J]ump.....	70
[Ctl_T]race.....	70
[Ctl_F]orce.....	70
[Ctl_eX]écute.....	71
[Alt_eX]écute.....	71
3.11 Points d'arrêt.....	72
[Ctl_B]reakpoint.....	72
[Alt_B]reakpoint.....	72
[Ctl_K]ill.....	74
[Ctl_D]étourne système.....	74
[Ctl_E]xception.....	74
[Help].....	75
3.12 Commandes diverses.....	77
[K]eep.....	77
[R]estore.....	77
[W]atch.....	77
[I]nterrupt Priority Level.....	77
[H]istory.....	78
[Alt_M]emory free.....	78
[F1].....	78
[F2].....	78
[Alt_P]rint.....	78
[P]rint.....	79

[E]value	79
[Alt_@]	79
[Ctl_Alt_Del]	80
[Sft_Ctl_Alt_Del]	80
[Sft_Alt_Help]	80
[O]utput	80
APPENDICES	81
Appendice 1 - Connaissances nécessaires à l'usage	
d'un débogueur	81
Le microprocesseur	81
L'assembleur	87
Les modes d'adressage	87
Les instructions du 68000	91
Notions de trace	92
Le mode Trace un peu plus en détail	93
Notions de point d'arrêt / exception	94
Notions d'interruption / exception	96
Gestion d'une interruption par le 68000	97
Prise en charge d'une interruption par le	
68000	99
Priorité des interruptions	100
Appendice 2 - Périphériques utilisables	101
Ecran	101
Disque (souple et dur)	101
La sortie RS232 (port RS232 - V24)	102
L'imprimante (port parallèle - Centronics)	103
Appendice 3 - L'évaluateur	105
Appendice 4 - Version cartouche. Changement de	
moniteur en cours de session	112
Appendice 5 - Index thématique des messages	
d'erreur	113
Appendice 7 - Table ASCII	118
Appendice 8 - Petite documentation de l'Atari ST,	
STE et TT	119
Liste des figures	136
Index général	139

1 Introduction

1.1 Contenu de la disquette

Adébog 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.

ADEBUGF.PRG version française.
RDEBUGF.PRG version française résidente.
AADEBUGF.ACC version française accessoire.

ADEBUG.PRG version anglaise.
RDEBUG.PRG version anglaise résidente.
AADEBUG.ACC version anglaise accessoire.

ADEBUG.VAR fichier de variables.
ADEBUG.MAC fichier de macro par défaut.

exemple.prg un programme d'exemple de débogage.
exemple.s le source de ce programme.

Un dossier ROS contenant

***.ro** les routines.
***.s** 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. Bien au contraire, le concept de débogueur est intimement lié à celui de programmation structurée.

Il est vrai que dans les temps préhistoriques de l'informatique (c'est à dire il y a environ 6 mois), les débogueurs s'appelaient des **moniteurs** et vous permettaient, grosso modo, d'afficher et de modifier le contenu de la mémoire de l'ordinateur. Les plus évolués d'entre eux permettant même de travailler directement en hexadécimal en lieu et place du binaire! En ces temps héroïques, il valait mieux passer une heure à analyser une ligne du programme source que deux semaines à la déboguer.

Aujourd'hui encore, on ne trouve sur certaines machines (et non des moindres) que ce type d'utilitaire. L'évolution décisive tient en fait au passage d'une présentation de type MS-DOS (saisie d'une commande sur une ligne, puis exécution) à un environnement plus interactif utilisant des fenêtres et des commandes d'une seule touche. Adébog est (comme d'autres outils), le fruit de cette évolution.

Ajoutons encore une dernière précision, afin d'éviter toute confusion: une nouvelle génération de débogueurs dits débogueurs source fait aujourd'hui son apparition. Il s'agit de débogueurs associés de façon très stricte à un langage de programmation, et fonctionnant un peu comme les instructions **TRON / TROFF** que l'on rencontre en basic. Ils interviennent en amont de toute compilation et ne permettent en aucun cas de suivre l'exécution d'un programme final. Adébog permet cette exécution même si le programme source n'est pas un programme 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 \$A0000** dans un programme chargé en mémoire avec Adébog. 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. Adébog sait relire cette section et, si votre langage le permet, il vous sera possible sous Adébog de visualiser ces noms pour simplifier le débogage. Ainsi le **GOSUB lecture** du basic sera affiché sous la forme **JSR lecture** par Adébog... et en lieu et place de l'adresse **\$A0000** vous trouverez le label **lecture**! En outre, comme Adébog connaît (grâce au fichier **ADEBUG.VAR**) toutes les routines et les variables de la ROM de l'Atari par leur petit nom, vous aurez l'impression en traçant 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 I 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 par le symbole 

les remarques simples par le symbole 

les remarques importantes par le symbole 

les remarques critiques par le symbole 

les combinaisons de touches entre crochets "[]" 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_0]

signifie touche Control et touche 0 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.

 L'état de la touche [Caps Lock] n'influe pas sur la prise en compte des raccourcis clavier.

2.2 Adébog, un programme n'utilisant pas GEM

Pour des raisons de rapidité d'utilisation et d'affichage, Adébog ne fait pas appel aux fonctions de l'interface graphique GEM ni à la ligne A. 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', Adébog, 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'Adébog utilise des fenêtres pour afficher ses données.

Cinq fenêtres peuvent être affichées simultanément, en quatre types de visualisation: **registres du 68000, désassemblage, 'dump' hexadécimal, ASCII.**

La fenêtre 1 affiche généralement le contenu des registres du 68000, bien qu'elle puisse aussi afficher la mémoire.

La fenêtre 2 affiche toujours le désassemblage d'une partie de la mémoire, contenant généralement le PC ("Program Counter", ou compteur ordinal).

Les fenêtres 3, 4 et 5 affichent une partie de la mémoire, dans n'importe quel mode (sauf registres du 68000).

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.

```

D0:*00FC9FC2 ^fK A0:*00000000 602E 0102 00FC 0030 0205 0956 0305 0956 IPLJ
D1: 00000000 A1:*00000000 602E 0102 00FC 0030 0205 0956 0305 0956 MC:
D2: 00000000 A2: 00000000 602E 0102 00FC 0030 0205 0956 0305 0956 TA:M
D3: 00000000 A3: 00000000 602E 0102 00FC 0030 0205 0956 0305 0956 TB:M
D4: 00000000 A4: 00000000 602E 0102 00FC 0030 0205 0956 0305 0956 TC:Y
D5: 00000000 A5: 00000000 602E 0102 00FC 0030 0205 0956 0305 0956 TD:M
D6: 00000000 A6: 0004750A 6000 2020 4172 662E 2020 206F 0004 202D
D7: 00000000 A7:*003F7FF8 0000 0000 0004 740A AAAA AAAA AAAA AAAA
SR:*0310 3X SSP:*0000755A 0101 0000 0000 0000 16C2 0000 168C 00FC
PC:*00FC0030 MOVE #52700,SR
    
```

00FC0030	◇ MOVE #52700,SR	00FC0030 46FC 2700 F01
00FC0034	RESET	00FC0034 4E70 0CB9 Mp1
00FC0036	CMPX.L #5FA52235F,cartridge	00FC0038 FA52 235F .R#
00FC0040	BNE.S \$FC004C	00FC003C 00FA 0000 .
00FC0042	LEA \$FC004C(PC),R6	00FC0040 660A 4DFA f0H.
00FC0044	JMP \$FA0004	
00FC004C	LEA \$FC0054(PC),R6	

00FC0030	◇ MOVE #52700,SR	; définition des EOU de 1 à ligne 8 MAX_X equ -5c MAX_Y equ -4 _X1 equ 30 _Y1 equ 40
00FC0034	RESET	
00FC0036	CMPX.L #5FA52235F,cartridge	
00FC0040	BNE.S \$FC004C	

Fenêtre 1, type registres

Fenêtre 5, type ASCII

Fenêtre 2, type désassemblage

Fenêtre 4, type désassemblage

Fenêtre 3, type hexadécimal

Figure 1:

Les cinq fenêtres d'Adébog et les quatre 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'Adébog qui en 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 servent à déplacer le curseur sur la ligne (les flèches haut et bas quand à elles 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_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 majuscule (souvent utile pour les noms de blocs).

[Ctl_L]ower fait l'inverse pour les majuscules.

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 singularités d'Adébog 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 fonctions supplémentaires dans Adébog, écrites par l'utilisateur et spécifiques à son application. Cette modularité d'Adébog permet à chacun de disposer de son propre débogueur, adapté à ses besoins et exigences.

Les différents types de variables seront amplement décrits dans le chapitre de références sur les variables (3.6).

2.5 Macros

Rappel

On note 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.



[Alt_A]

signifie touche Alternate et touche a pressées simultanément.

Un chiffre seul signifie la touche correspondant à ce chiffre sur le clavier (y compris le pavé numérique) enfoncée.



[Alt_1]

signifie touche Alternate et touche 1 pressées simultanément.

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



[F8]

signifie touche de fonction 8 enfoncée.

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'Adébog.

Chaque touche associée à une fonction est représentée par:

- Un signe ` (backquote).
- L'état des touches: **sft_** , **alt_** , **ctl_** (éventuellement).
- La lettre ou le chiffre correspondant à la touche du clavier utilisée, ou un mot la symbolisant.

 `ctl_z `ctl_u `down (qui signifie flèche bas) `sft_down `left `s ...

31 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, Adébog 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 références sur la configuration (3.9).

2.7 Ligne de commande à l'exécution

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

-**Soit** le nom d'un programme à charger (équivalente à la fonction [Ctl_L], voir le chapitre de références sur les opérations disque 3.8).



Du bureau:

Taper exemple dans la fenêtre de saisie des paramètres de l'application

Sous shell:

Taper adebug exemple

Lance Adébog en lui demandant de charger le programme **EXEMPLE.PRG**.

-**Soit** l'option **-b** suivie d'un nom de fichier binaire à charger (équivalente à la fonction [B], voir le chapitre de référence sur les opérations disque 3.8).



Au bureau:

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

Sous shell:

adebug -b data.bin

Lance Adébog en lui demandant de charger le fichier binaire **DATA.BIN**.

2.8 Usages d'un terminal

Il est pratique de pouvoir visualiser à la fois l'écran d'Adébog et celui du programme en cours de débogage. De plus, certains programmes peuvent utiliser l'écran de l'Atari de façon si complexe qu'il est impossible à Adébog de conserver un affichage propre (c'est par exemple le cas d'un programme de 'full-screen' (plein écran)). 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 géré par le programme en cours de débogage. Le nouveau 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 le ST 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 Version cartouche

Dans la mesure où Adébog réside en RAM, il est toujours susceptible d'être altéré par un programme qui ne tiendrait pas compte des réservations mémoire ou qui accéderait accidentellement à des zones réservées.

Afin de rendre Adébog quasi-invulnérable à ce genre de manipulations malencontreuses, une version en EPROMs, enfichable sur le port cartouche de l'Atari, a été mise au point. Elle permet en outre un chargement plus rapide.

Vous trouverez en appendice 4 de plus amples informations.

2.10 Changement de moniteur en cours de session

Il est très désagréable de devoir faire Reset pour changer de moniteur/résolution, ou d'utiliser un éditeur sur moniteur couleur. Avec l'interface de changement de moniteurs, Adébog (lancé en monochrome) peut commuter sur l'écran couleur, puis ensuite revenir en haute résolution (pour continuer à éditer un source par exemple). Une fois le drapeau **échangeur électronique** positionné dans les préférences, appuyer sur [F8] passe en basse résolution, sur [F9] passe en moyenne résolution, et [F10] permet de revenir en haute résolution.

☞ Dans ce mode, si l'on quitte Adébog en couleur, il revient automatiquement en haute résolution.

Cette possibilité de changer de moniteur en cours de session peut permettre, par exemple, de déboguer en monochrome un programme fonctionnant normalement en couleur, avantage appréciable en particulier lors du développement de jeux.

Vous trouverez en appendice 4 de plus amples informations.

2.11 Exemple d'utilisation

Et maintenant, un petit exemple d'utilisation.

Lancez Adébog, puis à la demande:

Charger prg:

entrez le nom de programme:

exemple

Adébog ajoutera lui-même l'extension appropriée (dans ce cas, .PRG).

A la demande:

Ligne de commande:

tapez simplement [Return], ce programme d'exemple ne l'utilisant pas.

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:

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 étoile (*). 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 à nouveau descendue. Mais c'est maintenant une flèche vers le bas. En effet, l'instruction en face de la flèche est une instruction de branchement (BSR) à une adresse supérieure à l'actuel PC. La flèche indique donc la direction du branchement.

De nouveau grâce à [Ctl_Z], le branchement est effectué. La flèche indique de nouveau la droite, car la prochaine instruction ne fera pas de branchement. Dans la fenêtre 1, 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 [Ctl_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 [Ctl_Z]. En faisant [Ctl_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 **Lancé et stoppé.** est affiché. Il ne reste plus qu'à effectuer le RTS (avec [Ctl_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 [Ctl_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

3 Références

3.1 Liste thématique des fonctions

3.3 Fenêtres

[Alt_Z]oom agrandir la taille de la fenêtre courante

[Tab] changer la fenêtre courante

[Alt_S]éparer et regrouper les fenêtres

[Alt_T]ype de visualisation dans une fenêtre

[Alt_1~5] aller sur la fenêtre 1~5

[Ctl_0~7] mettre l'adresse de la fenêtre courante à A0~7

[Ctl_8] mettre l'adresse de la fenêtre courante à SSP

[Ctl_9] mettre l'adresse de la fenêtre courante à PC

[Sft_Ctl_0~7] mettre l'adresse de la fenêtre courante à {A0~7}

[Sft_Ctl_8] mettre l'adresse de la fenêtre courante à {SSP}

[Sft_Ctl_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 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

3.4 Gestion et édition mémoire

[Alt_E]dition de fenêtre

[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_N]ext chercher en arrière l'expression

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. Ce dernier est donc terminé (il suffit de faire [Ctl_Z] deux fois pour le vérifier).

;exemple.s exemple de programme à tracer pour Adébog.

opt d+ ; pour avoir les symboles de débogage.

TEXT

Start:

```

move.w    #$1234,d0
moveq    #2,d1
bsr      routin_1
move.w    #$4321,d0
moveq    #-2,d1
bsr      routin_1
moveq    #-2,d0
move.w    #$4321,d1
bsr      routin_2
clr.w    -(sp)
trap     #1

routin_1:
nop
dbf     d0,routin_1
rts

routin_2:
nop
dbf     d1,routin_2
rts

END
```

3.5 Macros

[M]acro menu des macros

3.6 Variables

[Alt_V]variable créer variable

ADEBUG.VAR fichier d'initialisation des variables

[Help] liste des tampons

[L]iste des variables

3.7 Ecran

[V]oir l'écran logique

[Ctl_Alt_I]nverser la palette de couleurs d'Adébog

[Ctl_O]utput changer Adébog de résolution en couleur

[F8] passer en basse résolution

[F9] passer en moyenne résolution

[F10] passer en haute résolution

3.8 Opérations disque

[D]irectory changer de répertoire courant

[Alt_D]irectory afficher le répertoire courant et son contenu

[Ctl_L]oad charger programme

[B]inary charger fichier binaire

[Alt_A]SCII charger fichier ASCII

[S]auver en binaire une partie de la mémoire

3.9 Configurations

[Ctl_P]références

3.10 Opérations de contrôle de flux du programme

- [Ctl_C] terminer Adébog ou le programme en cours de débogage
- [Ctl_Z] tracer une instruction
- [Ctl_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 paramétrable
- [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.
- [Ctl_F]orcer le branchement
- [Ctl_eX]écuter routine
- [Alt_eX]écuter retour

3.11 Points d'arrêt

- [Ctl_B]reakpoint mettre un point d'arrêt sur la fenêtre courante
- [Alt_B]reakpoint mettre un point d'arrêt configurable
- [Ctl_K]ill détruire tous les points d'arrêts présents
- [Ctl_D]étourner un TRAP
- [Ctl_E]xception installer toutes ou certaines exceptions
- [Help] liste des points d'arrêt

3.12 Commandes diverses

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

[R]estaurer la valeur de tous les registres

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

[I]nterrupt Priority Level régler l'IPL

[H]istorique des instructions

[Alt_M]émoire libre

[F1] poser marque

[F2] échanger curseur et marque

[Alt_P]rint imprimer le contenu de la fenêtre courante

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

[E]valuer une expression

[Alt_@]robace

[Ctl_Alt_Del]ete provoquer un reset à chaud

[Sft_Ctl_Alt_Del]ete provoquer un reset à froid

[Sft_Alt_Help] arrêter le programme en cours d'exécution

[O]utput afficher sur écran ou sur RS232

3.2 Liste alphabétique des fonctions

[Tab] changer la 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

[Help] liste des points d'arrêt

[F1] poser marque

[F2] échanger curseur et marque

[F8] passer en basse résolution

[F9] passer en moyenne résolution

[F10] passer en haute résolution

- [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 l'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 RS232
- [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 paramétrable
- [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_@]robace
[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]iter 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_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_0~7] mettre l'adresse de la fenêtre courante à A0~7
- [Ctl_8] mettre l'adresse de la fenêtre courante à SSP
- [Ctl_9] mettre l'adresse de la fenêtre courante à PC
- [Sft_Ctl_0~7] mettre l'adresse de la fenêtre courante à [A0~7]
- [Sft_Ctl_8] mettre l'adresse de la fenêtre courante à [SSP]
- [Sft_Ctl_9] mettre l'adresse de la fenêtre courante à [PC]
- [Ctl_Flèche haut] changer la hauteur de la fenêtre courante
- [Ctl_Flèche bas] changer 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 Adébug 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_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_O]utput changer Adébug de résolution en couleur
- [Ctl_P]références changer et sauver les préférences
- [Ctl_R]un lancer l'exécution
- [Ctl_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_eX]écuter une routine
- [Ctl_Z] tracer une instruction

- [Ctl_Alt_Del]ete provoquer un reset à chaud
- [Ctl_Alt_I]nverser la palette de couleurs d'Adebug
- [Sft_Alt_Hclp] arrêter le programme en cours d'exécution
- [Sft_Ctl_Alt_Del]ete provoquer un reset à froid

3.3 Fenêtres

Les **fenêtres** d'Adébog 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.

☞ Il est impossible d'exécuter cette commande sur la fenêtre 1 en mode Registres.

00FC0030	◇ MOVE #52700,SR
00FC0034	RESET
00FC0036	SUBA.L A5,A5
00FC0038	CMPI.L #5FA52235F,cartridge
00FC0042	BNE.S \$FC004E
00FC0044	LEA \$FC004E(PC),A6
00FC0048	JMP \$FA0004
00FC004E	LEA \$FC0056(PC),A6
00FC0052	BRA \$FC066A
00FC0056	BNE.S \$FC005E
00FC0058	MOVE.B memctrl(A5),-\$7FFF(A5)
00FC005E	CMPI.L #531415926,resvalid(A5)
00FC0066	BNE.S \$FC0080
00FC0068	MOVE.L resvector(A5),D0
00FC006C	TST.B resvector(A5)
00FC0070	BNE.S \$FC0080
00FC0072	BTST #0,D0
00FC0076	BNE.S \$FC0080
00FC0078	MOVEA.L D0,A0
00FC007A	LEA \$FC005E(PC),A6
00FC007E	JMP (A0)
00FC0080	SUBA.L A5,A5
00FC0082	LEA -\$7000(A5),A0

Figure 2:

Une fenêtre en mode plein écran ([Alt_Z]oom de fenêtre).

[Tab]

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

[Alt_S]épare/Regroupe fenêtres

Séparer une fenêtre en deux ou regrouper deux fenêtres en une.

Sur la fenêtre 1:

La fait disparaître de l'écran. Pour la faire réapparaître, appuyer sur [Alt_1].

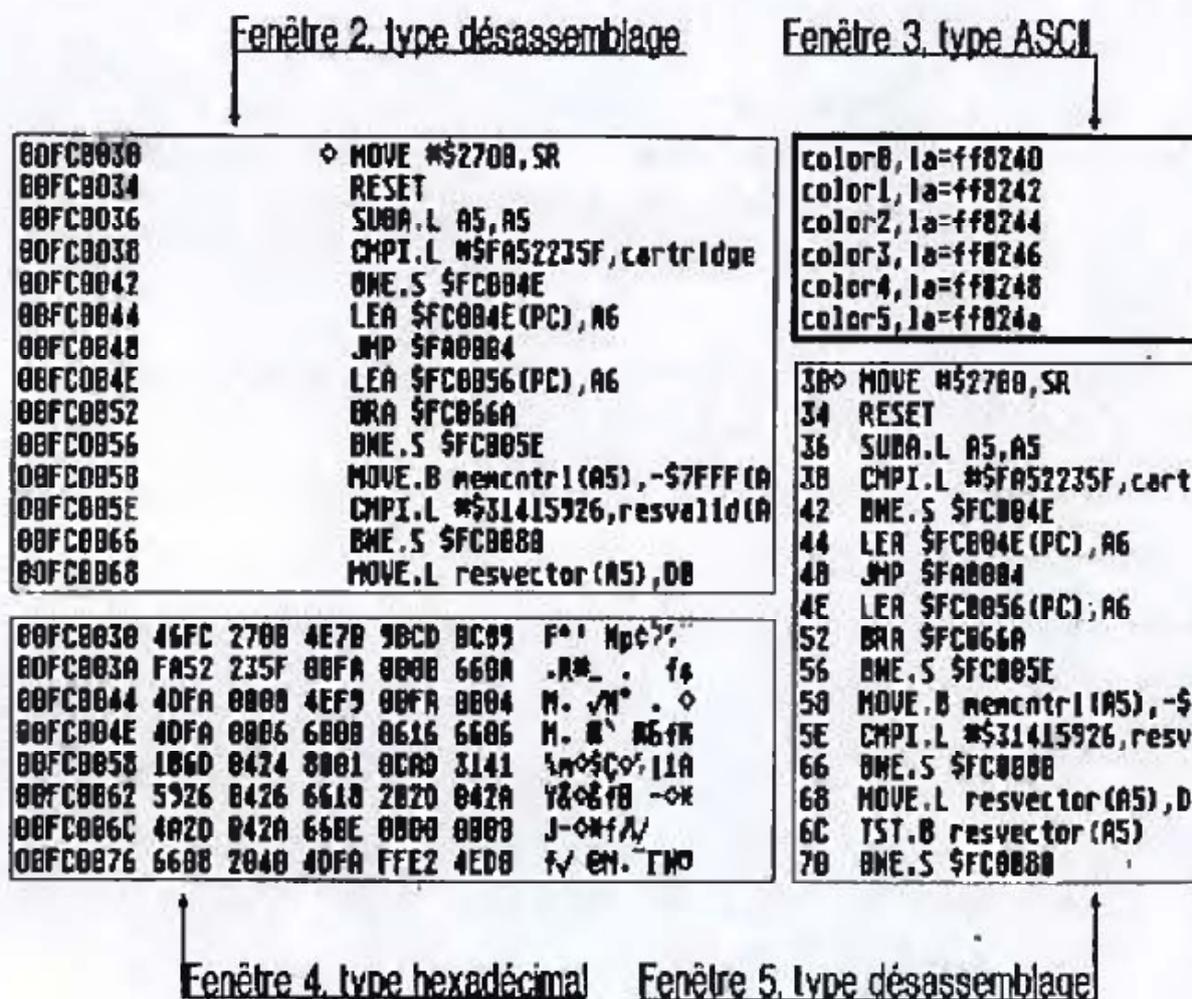


Figure 3:

Le mode quatre fenêtres par élimination de la fenêtre 1.

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.

☞ En basse résolution, cette commande n'a pas d'effet.

[Alt_T] type de visualisation

Changer le **type de visualisation** de la fenêtre courante.

Quatre types de visualisation sont possibles à l'intérieur d'une fenêtre: (voir figure 1)

Registres du 68000.

Désassemblage.

"Dump" hexadécimal.

"Dump" ASCII.

☞ La fenêtre 1 ne peut pas être de type désassemblage et la fenêtre 2 ne peut changer de type.

[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.

ICtl_0-71

Sur la fenêtre 1 (en visualisation de type registres):

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

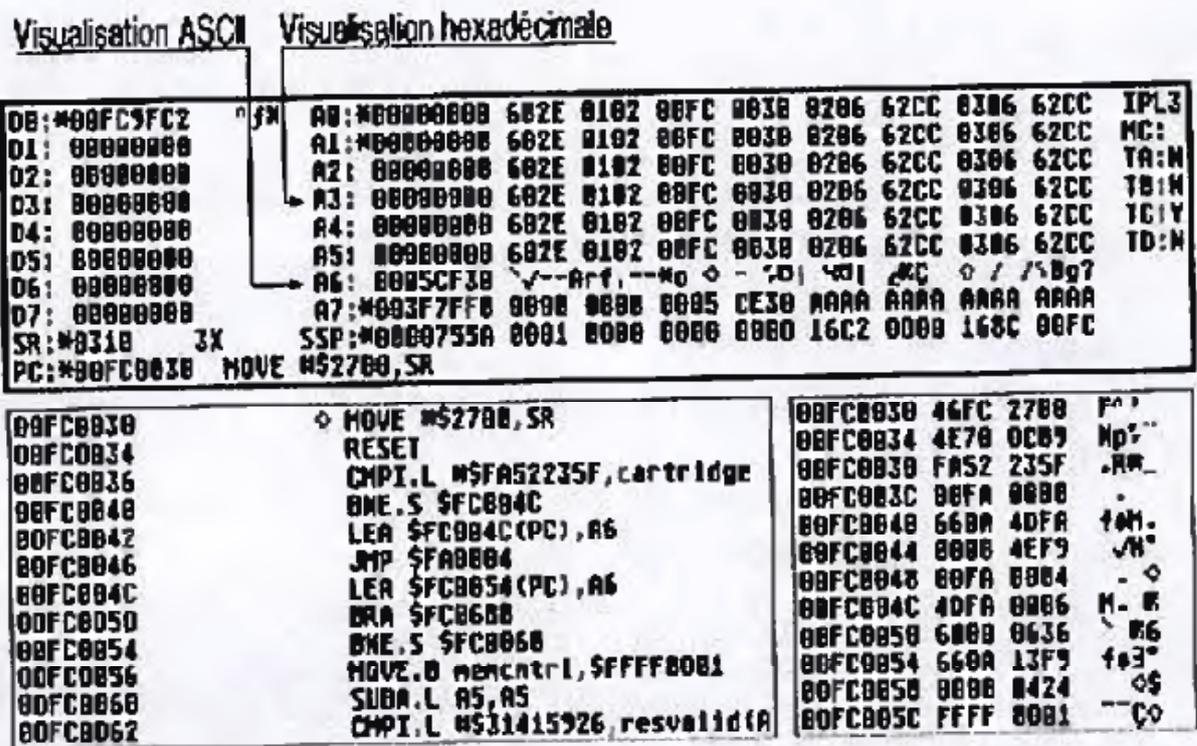


Figure 4:

Le mode visualisation ASCII du contenu des registres dans la fenêtre 1.

Dans tous les autres cas:
positionner la fenêtre courante sur le registre **A0~A7**.

[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).

[Sft_Ctl_0-7]

Positionner la fenêtre courante sur le contenu du registre **A0~A7**.

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

[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]

Remonter l'adresse d'affichage de la fenêtre courante d'un ou de deux octets (selon le type de la fenêtre courante).

[Flèche droite]

Redescendre l'adresse d'affichage de la fenêtre courante d'un ou de deux octets (selon le type de la fenêtre courante).

[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 haut]

Si la fenêtre courante est la fenêtre 4 ou 5, ou bien 2 ou 3 avec respectivement la fenêtre 4 ou 5 créée (par [Alt_S]), changer la taille de la fenêtre courante

[Ctl flèche bas]

Si la fenêtre courante est la fenêtre 4 ou 5, ou bien 2 ou 3 avec respectivement la fenêtre 4 ou 5 créée (par [Alt_S]), changer la taille de la fenêtre courante

[A] Adresse de fenêtre

Modifier l'adresse de base de la fenêtre courante. Toutes les expressions comprises par l'évaluateur sont valides.

 **[A] Expression <e>**: $(5+7)/4$
met la fenêtre courante à l'adresse 3.

[Alt_L]ock

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

 [Alt_L] Lie expression <e>: (4BA)
lie la fenêtre courante avec le contenu de l'adresse \$4ba
(compteur 200 Hz).

3.4 Gestion et édition mémoire

[Alt_E]dition de fenêtre

Sur la fenêtre 1 en type **Registres du 68000**;

Passer en mode édition du **SR, Timer A, Timer B, Timer C, Timer 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.

Registre d'état (SR)

Edition du Timer A

Timer B

Timer C

Timer D

D0: *00FC9FC2	^ fM	A0: *00000000	602E 0102 00FC 0030 0204 6154 0304 6154	IPL3
D1: 00000000		A1: *00000000	602E 0102 00FC 0030 0204 6154 0304 6154	MC:
D2: 00000000		A2: 00000000	602E 0102 00FC 0030 0204 6154 0304 6154	TD:N
D3: 00000000		A3: 00000000	602E 0102 00FC 0030 0204 6154 0304 6154	TB:M
D4: 00000000		A4: 00000000	.00 ^ 000aT00aT00aT00aTR0aTR0aT00aT00a	TC:Y
D5: 00000000		A5: 00000000	.00 ^ 000aT00aT00aT00aTR0aTR0aT00aT00a	TD:H
D6: 00000000		A6: 0093C0C8	✓--Arf.--k0 0 - %01 401 cR0 0 / /%Bg?	
D7: 00000000		A7: *003F7FF8	0000 0000 0093 CCC0 AAAA AAAA AAAA AAAA	
SR: *0310	JX	SSP: *0000755A	0000 0000 0000 0000 16C2 0000 168C 00FC	
PC: *00FC0030	MOVE #52700,SR			

00FC0030	◇ MOVE #52700,SR	00FC0030	46FC	2700	F01
00FC0034	RESET	00FC0034	4E70	0C09	Np1
00FC0036	CMPI.L #5FA52235F,cartridge	00FC0038	FA52	235F	.RH
00FC0040	BNE.S \$FC004C	00FC003C	00FA	0000	.
00FC0042	LEA \$FC004C(PC),A6	00FC0040	660A	4DFA	faM.
00FC0046	JMP \$FA0004	00FC0044	0000	4EF9	√M*
00FC004C	LEA \$FC0054(PC),A6	00FC0048	00FA	0004	. . 0
00FC0050	BRA \$FC0060	00FC004C	4DFA	0006	M. R
00FC0054	BNE.S \$FC0060	00FC0050	6000	0636	\ 06
00FC0056	MOVE.B memcntr1,\$FFFF0001	00FC0054	660A	13F9	fa3*
00FC0060	SUBA.L A5,A5	00FC0058	0000	0424	05
00FC0062	CMPI.L #531415926,resvalid(A	00FC005C	FFFF	0001	—00

Figure 5: Le mode édition du SR et des timers dans la fenêtre 1 ([Alt_E]).

Sur une fenêtre de 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.

```

D0: *00FC9FC2  *fx  A0: *00000000 602E 0102 00FC 0030 0206 A2DC 0306 A2DC IPL3
D1: 00000000  A1: *00000000 602E 0102 00FC 0030 0206 A2DC 0306 A2DC MC:
D2: 00000000  A2: 00000000 602E 0102 00FC 0030 0206 A2DC 0306 A2DC TA: M
D3: 00000000  A3: 00000000 602E 0102 00FC 0030 0206 A2DC 0306 A2DC TB: M
D4: 00000000  A4: 00000000 602E 0102 00FC 0030 0206 A2DC 0306 A2DC TC: Y
D5: 00000000  A5: 00000000 602E 0102 00FC 0030 0206 A2DC 0306 A2DC TD: M
D6: 00000000  A6: 00060F40 6000 2020 4172 662E 2020 2A6F 0004 2020
D7: 00000000  A7: *003F7FF0 0000 0000 0000 0E40 AAAA AAAA AAAA AAAA
SR: *0310 IX SSP: *00007550 0001 0000 0000 0000 16C2 0000 168C 00FC
PC: *00FC0030 MOVE #52700, SR
    
```

```

00FC0030  ◊ MOVE #52700, SR
00FC0034  RESET
00FC0036  CMPI.L #5FA52235F, cartridge
00FC0040  BNE.S $FC004C
00FC0042  LEA $FC004C(PC), A6
00FC0046  JMP $F00004
00FC004C  LEA $FC0054(PC), A6
00FC0050  BRA $FC0680
00FC0054  BNE.S $FC0060
00FC0056  MOVE.B memcntrl, $FFFF0001
00FC0060  SUBA.L A5, A5
00FC0062  CMPI.L #531415926, resvalid(A
    
```

Adresse <E>: 300000

```

00FC0030 46FC 2700 F000
00FC0034 4E70 0C09 Np00
    
```

```

00300000 4564 6974 Edit
00300004 696F 6E20 ion
00300008 6465 206C de l
0030000C 6120 6002 → né
00300010 606F 6972 noir
00300014 652E 2E2E e...
00300018 0000 0000
0030001C 0000 0000
00300020 0000 0000
    
```

Edition de la mémoire

Figure 6

Le mode édition de mémoire en hexadécimal/ASCII ([Alt_E]).

[C]opier

Copier un bloc-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.

 **[C] Copier <source>, <longueur>, <destination>: 8,ff,1000**
copie \$ff octets de l'adresse \$8 à l'adresse \$1000.

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

[F]ill

Remplir un bloc-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] 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] 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 d'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] Long 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'Adébog. Chaque touche associée à une fonction est représentée par le signe ``` (**backquote**).

L'état des touches: `sft_`, `alt_`, `ctl_` (s'il y a lieu).
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.

 ``ct_b est valide.`
``ct`
`_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)racier A)rrière P)asser C)lr

Enregistrer

Lancer l'enregistrement de macro.

Tracier

Lancer l'enregistrement d'une seule 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)racier A)rrière P)asser V)oir D)ébut

Jouer

Lancer l'exécution de la macro en mémoire.

Tracier

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.
[Sft_Flèche bas] ou [Espace]:	descendre d'une page.
[Sft_Flèche haut] ou [Sft_Espace]:	remonter d'une page.
[Home]:	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.

A ce moment, si la fenêtre 1 est en mode visualisation registres, le drapeau **MC** indique l'enregistrement (avec la lettre **R** pour **Record** ou Enregistrement).

Dorénavant, chacune des vos actions dans Adébog 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 textes 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 ° (degré), 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_?. Continuer 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. Continuer 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.

°l <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.



°l boucle
`ctl z
°b d0!=0 boucle

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

3.6 Variables

{Alt_V}variable

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

<nom>, <type>=<expression>.

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

***, /, +, -, ^, ~, |, &, \$, \, @, (,), [,], .., {, }, ..., =, <, >, !**

 Les symboles réservés sont automatiquement éliminés.

Une variable peut être de cinq 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).

 mfp,la=FFFFFFA01

La VAR mfp est de type LA, et sa valeur est \$FFFFFFA01.

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 200 Hz (\$4ba).

BL: Bloc

Tout fichier chargé en mémoire par Adébog (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 en majuscules) 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 suffit de 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 superviseur, avec des piles utilisateur et superviseur qui lui sont propres.

 linea,ro='linea.ro

Initialise la RO linea par le chargement du fichier LINEA.RO

Afin de retourner dans Adébog, elle doit se terminer non par un Pterm, mais par un RTS en mode superviseur (la pile ne doit donc pas être décalée). Cette routine est libre d'effectuer n'importe quelle opération à l'intérieur d'elle même.

Elle peut néanmoins communiquer avec Adébog. Elle reçoit à l'exécution dans le registre AO l'adresse d'une structure de communication (source fourni sur la disquette).

ROSTRUCT.S

Structure de communication entre une routine (RO) et Adébog.

offset 0

ro_struct:

 ;numéro de version d'Adébog
v_number:
 ds.w 1

 ;numéro de version du TOS
tos_number:
 ds.w 1

 ;adresse de la fenêtre W1
window_1:
 ds.l 1

 ;adresse de la fenêtre W2
window_2:
 ds.l 1

 ;adresse de la fenêtre W3
window_3:
 ds.l 1

 ;adresse de la fenêtre W4
window_4:
 ds.l 1

 ;adresse de la fenêtre W5
window_5:
 ds.l 1

 ;adresse de la section TEXT du programme en cours de débogage
TEXT:
 ds.l 1

 ;adresse de la section DATA du programme en cours de débogage
DATA:
 ds.l 1

 ;adresse de la section BSS du programme en cours de débogage
BSS:
 ds.l 1

 ;adresse de la fin du programme en cours de débogage
END:
 ds.l 1

 ;valeur des registres du programme en cours de débogage
registers:
d0_reg:
 ds.l 1

d1_reg:
ds.l 1

d2_reg:
ds.l 1

d3_reg:
ds.l 1

d4_reg:
ds.l 1

d5_reg:
ds.l 1

d6_reg:
ds.l 1

d7_reg:
ds.l 1

a0_reg:
ds.l 1

a1_reg:
ds.l 1

a2_reg:
ds.l 1

a3_reg:
ds.l 1

a4_reg:
ds.l 1

a5_reg:
ds.l 1

a6_reg:
ds.l 1

a7_reg:
ds.l 1

ssp_reg:
ds.l 1

Adébog _____ @

sr_reg:
 ds.w 1

pc_reg:
 ds.l 1

 ;résolution du programme en cours de débogage

reso:
 ds.w 1

 ;adresse de la basepage du programme en cours de débogage

basepage_addr:
 ds.l 1

 ;adresse de la routine en cours d'exécution

ro_addr:
 ds.l 1

 ;adresse de l'écran du programme en cours de débogage (écran
 logique)

logic_screen_addr:
 ds.l 1

 ;contient l'adresse d'une chaîne à afficher au retour de la RO (ou
 sinon 0)

string_addr:
 ds.l 1

 ;réservé pour extensions futures

reserved:
 ds.l 2

 ;drapeau signalant à Adébog de réinstaller toutes les exceptions

reput_exc:
 ds.b 1

 ;drapeau signalant à Adébog de passer en IPL7

IPL7:
 ds.b 1

 ;drapeau contrôlant la marche ou l'arrêt du timer A

timera:
 ds.b 1

 ;drapeau contrôlant la marche ou l'arrêt du timer B

timerb:
 ds.b 1

 ;drapeau contrôlant la marche ou l'arrêt du timer C

timerc:

ds.b 1

;drapeau contrôlant la marche ou l'arrêt du timer D

timerd:

ds.b 1

;drapeau signalant à Adébog de redessiner son écran

redraw_screen:

ds.b 1

;drapeau signalant à Adébog de passer en sortie RS232

rs232_output:

ds.b 1

;réservé pour extensions futures

reserved2:

ds.b 1

De plus, la routine doit pouvoir indiquer quelque chose à l'utilisateur. Ainsi, la valeur renvoyée dans le registre **D0** en sortie de la RO 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 TEXT. Elles permettent de créer des symboles relatifs au début du programme à déboguer avant que celui-ci ne soit chargé.

 basepage,ex=-100

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

ADEBUG.VAR:

Fichier ASCII contenant les définitions de VARs que l'on désire initialiser au chargement d'Adébog. 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 ligne vide.

[Help]

Visualiser les informations générales et les points d'arrêt.

Cette fonction comporte trois parties à l'affichage:

Première partie: L'adresse et la longueur des quatre sections du programme en cours de débogage.

Deuxième partie: Les nombres maximum et courant des points d'arrêt, des blocs, et des instructions enregistrées dans l'historique (STO) ainsi que les tailles maximum et courante des tampons VAR, LA, LR, BL, EX, MAC, HIS.

Troisième partie:

(Voir chapitre 3.11 sur les points d'arrêt.)

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

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

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

Adebug v 1.00 (c) 1990 Brainstorm. (L. Chenla, A. Lenaresquier, R. Lenoire).

Nom	Adresse	Long.	N° Adresse	Type Instruction
TEXT	0006792C	00000032		
DATA	0006795E	00000000		
BSS	0006795E	00000000		
SYM	0006795E	0000002A		

Adresse et taille des sections TEXT, DATA, BSS et Symboles

Nom	Maximum	Courant	Description
BKH	256	0	Nombre de points d'arrêt
BLN	30	17	Nombre de blocs
STH	8	3	
VAR	10000	4710	Nombre d'entrées dans l'historique des instructions
LA	10000	3705	
LR	1000	0	Tampon de variables
BL	15000	8502	Tampon de labels absolus
EX	100	0	Tampon de labels relatifs
MAC	5000	10	Tampon de blocs
HIS	200	40	Tampon de variables existant
			Tampon de macro
			Tampon d'historique

Figure 7:

Liste des tailles courantes et maximum des tampons de variables et points d'arrêt.

[L]iste des variables

Afficher toutes les variables.

Les variables sont affichées dans le même format que lors de leur création (voir [Alt_V]): **<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.

EX: Leur valeur ajoutée à celle de la section TEXT du programme en cours de débogage.

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

```

;page n°5
scr, la=FFFA27
ucr, la=FFFA29
csr, la=FFFA2B
tsr, la=FFFA2D
udr, la=FFFA2F
keyctl, la=FFFCB8
keybd, la=FFFCB2
nidictl, la=FFFCB4
nidi, la=FFFCB6
vblvec, la=78
timercvec, la=114
ikbdvec, la=118
hbvec, la=120
proc_lives, la=388
proc_regs, la=384
proc_pc, la=3C4
proc_esp, la=3C8
proc_stk, la=3CC
etv_timer, la=488
etv_critc, la=484
etv_tern, la=488
etv_xtra, la=48C

```

Contenu de la variable

Type de la variable

Nom de la variable

Figure 8:
Liste des variables (IL).

3.7 Ecran

[V]oir

Inverser l'**écran logique** et l'**écran physique**. Toutes les fonctions d'Adébog restent disponibles quelque soit l'écran visualisé.

[Ctl_Alt_I]nverse

Inverser la **couleur d'encre** et la **couleur de fond** d'Adébog.

[Ctl_O]utput

Basculer l'écran d'Adébog de la **moyenne résolution** à la **basse résolution** et réciproquement.

☞ Cette fonction n'a pas d'effet sur moniteur monochrome.

Les trois fonctions suivantes ne sont utilisables qu'avec l'interface de changement de moniteur en ayant positionné le **drapeau** correspondant dans les préférences.

[F8]

Passer en **basse résolution**.

[F9]

Passer en **moyenne résolution**.

[F10]

Passer en **haute résolution**.

3.8 Opérations 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 (* ?).

 [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 ne peut en contenir la fenêtre, appuyer sur une touche donne accès à la suite du répertoire.

La touche [Esc] permet de revenir à l'écran principal. La touche [P] permet d'imprimer le contenu du répertoire courant.

[Ctl_L]oad program

Charger un **fichier relogeable** ou **programme**.

Cette fonction demande un nom de **programme exécutable**. Le nom peut comprendre des caractères joker (* ?) et ne pas comprendre d'extension. Dans ce cas, si le nom sans extension n'est pas trouvé, les extensions **.prg**, **.tos**, **.app**, **.ttp** lui seront successivement rajoutées. Une ligne de commande optionnelle est alors demandée puis le programme correspondant est chargé et relogé. Ensuite, les symboles de débogage sont analysés et enregistrés. Les formats de symbole reconnus sont les formats DRI (symbole de 8 caractères) et HiSoft Extended Debug (8 ou 22 caractères) sans qu'il soit besoin de le préciser.

 [Ctl_L] **Charger prg:** exemple charge le programme EXEMPLE.PRG.

 Une fois qu'un programme a été chargé avec cette commande, celle-ci est impossible à 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, Adébog lui allouant en mémoire la taille qu'il a sur disque et le chargeant à l'adresse fournie par l'allocation.



[B] Charger binaire: data.bin

charge le fichier DATA.BIN à l'adresse résultant de l'allocation mémoire.

L'autre méthode demande, en sus du nom de fichier, une adresse où le charger en mémoire. Il n'y a donc pas d'allocation mémoire.



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

charge le fichier SCREEN.BIN à l'adresse contenue dans \$44e (adresse de l'écran logique).

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 en majuscules.



Le nom de fichier se trouve débarrassé des caractères interdits tels que . (point), # (dièse), !, etc ...

[Alt_A]ASCII

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**. En conséquence, cette fonction n'est pas disponible sur la fenêtre 2 (voir le chapitre sur les fenêtres).

 [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.

 [S] **Sauver binaire:** screen.bin, {44E},\32000 sauve les 32000 octets de l'écran logique sur disque sous le nom SCREEN.BIN.

3.9 Configurations

Pour presque chacun des types de **VAR** expliqués dans le chapitre sur les variables, correspondent 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 utilisant 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 aux 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 celui-ci.

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'Adébog.

[Ctl_P]références

Modification et sauvegarde des préférences d'Adébog.

Liste des préférences, et de leur valeur par défaut:

Valeur de tabulation pour l'affichage de type ASCII dans les fenêtres: Défaut: 5

Longueur du format des symboles du programme à charger:
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 par défaut au lancement d'Adébog: Défaut: 3

Vitesse de la RS232: (voir l'explication dans l'appendice 2 sur la RS232) Défaut: 2. (Soit 4800 bauds, pour le minitel 1B)

Parité de la RS232: (voir l'explication dans l'appendice 2 sur la RS232) Défaut: \$be

Nombre maximum de points d'arrêt: Défaut: \256

Nombre maximum de blocs: Défaut: \30

Nombre d'entrées dans l'historique des instructions (STO):
Défaut: 8

Taille du tampon de variables (VAR): Défaut: \10000

Taille du tampon d'étiquettes (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

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

Réinstalle l'adresse correspondant au mode Trace avant de tracer: Défaut: Oui

Commute l'écran d'Adébog et l'écran logique en cas de [Ctl_A]: Défaut: Oui

Commute l'écran d'Adébog et l'écran logique en cas de [Ctl_R]: Défaut: Oui

Commute l'écran d'Adébog et l'écran logique en cas d'exception Line A-F, Division par zéro, Chk, 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

Affiche les instructions 68010: Défaut: Oui

Autorise le temps réel: Défaut: Oui

Autorise le temps réel uniquement dans la fenêtre courante:
Défaut: Non

Autorise le lien des fenêtres en temps réel: Défaut: Non

Signale si la sortie RS232 doit être effectuée en émulation minitel: Défaut: Oui

Commute l'affichage d'Adébog en sortie RS232 à son lancement: Défaut: Non

Permet d'utiliser les fonctions [F8] [F9] [F10] en combinaison avec le commutateur électronique de moniteur. Défaut: Non

3.10 Opérations de contrôle de flux du programme ,

[Ctl_C]

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

☛ 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.

[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_Skip]

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

Emule 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:

BSR, JSR, BRA, JMP, RTS, RTE, 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'Adébog, 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 choisi.



Il est préférable d'avoir l'IPL d'Adébog identique à celui du programme en cours de débogage si l'on souhaite un arrêt immédiat du programme avec [Sft_Sft].

 En mode

Voir écran logique

le clignotement du premier pixel en haut et à gauche de l'écran indique que l'exécution continue.

[U]ntil

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



[U] Lancer jusqu'à: 3f342

Lance l'exécution jusqu'à l'adresse \$3f342.

[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, JSR, JMP, Bcc, DBcc).

[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.)

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 **visualisation désassemblage**.

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 évalué 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 , correspondantes.

 [Alt_B] Point d'arrêt: ,,1

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

[Ctl_K]III

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'Adébog).

[Ctl_D]étourne système

Détourne une **exception TRAP** et permet ainsi d'arrêter l'exécution en cas d'appel correspondant au numéro donné en premier paramètre.

Comme deuxième paramètre, il est possible de spécifier le numéro de fonction pour lequel l'exécution doit être arrêtée. Tant que le numéro de fonction n'est pas identique à ce dernier, le TRAP originel est appelé et l'exécution continue. En cas d'égalité, cette dernière est arrêtée.

 Le numéro de fonction considéré est pris en **mot**, comme c'est le cas actuellement pour les fonctions système de l'ATARI. Pour enlever le détournement, il suffit de rappeler [Ctl_D] et de taper une ligne vide.

 **[Ctl_D] Détourner <trap n°, fonction n°> : 1,3d**

Détourne la fonction \$3d (Fopen) du trap #1 (Gemdos).

[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]).

[Help]

Cette fonction comprend à l'affichage 3 parties. Les deux premières parties ne concernent pas les points d'arrêt. La troisième partie permet de voir et de manipuler la **liste des points d'arrêt**. Chacun de ceux-ci se présente comme suit:

<numéro>, <adresse>, <type>, <numéro de vecteur>, <instruction [expression]>.

<numéro> est le numéro du point d'arrêt.

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

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

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

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

[expression] est l'expression du point d'arrêt.

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

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

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

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

[Sft_Flèche bas] ou [Espace]: descendre dans la liste d'une page.

[Sft_Flèche haut] ou [Sft_Espace]: remonter dans la liste d'une page.

[Home]: amène en début de liste.

[Sft_Home]: amène en fin de liste.

[P]: impression de la liste.

[S]: sauvegarde de la liste.

Points d'arrêt permanents			Liste des points d'arrêt			Expression à évaluer		
Rdebug v 1.00 (c) 1998 Brainstorm. (L. Chenje, A. Lemaesquier, R. Lemoine).								
Nom	Adresse	Lang.	N°	Adresse	Type	Instruction		
TEXT	0006792C	00000032	1	0006792C	20	MOVE.W #51234,00 [-1]		
DATA	0006795E	00000000	2	00067932	20	BSR routin_1 [-1]		
BSS	0006795E	00000000	3	0006793A	20	MOVEQ #-2,01 [d0==0]		
SYM	0006795E	0000002A	4	00067940	P 26	MOVEQ #-2,00 [a7==ssp]		
			5	00067946	20	BSR routin_2 [-1]		
			6	0006794A	20	CLR.W -(A7) [-1]		
			7	0006794E	20	HOP [-1]		
			8	00067956	P 20	HOP [-1]		
			9	00067962	20	MOVEQ #0,D2 [-1]		
			A	00067958	20	DBF 01,routin_2 [-1]		
Nom	Maximum	Courant						
OKM	256	10						
BLM	30	17						
STM	0	3						
VAR	10000	4710						
LA	10000	3705						
LR	1000	0						
BL	15000	0502						
EX	100	0						
MRC	5000	10						
HIS	200	40						

Point d'arrêt n°8 mis.

Figure 9:

Liste des points d'arrêt ((Help)).

3.12 Commandes diverses

[K]eep

Mémoriser la valeur de tous les **registres du 68000** afin de les restituer avec [R]estore.

[R]estore

Restaurer la valeur de tous les **registres du 68000** (si ceux-ci ont été sauvés avec [K]), et rafraîchit l'écran.

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

[W]atch

Si le PC pointe sur une instruction de saut, permet de visualiser dans la fenêtre 2 l'adresse de destination. Cette évaluation accepte tous les modes d'adressage autorisés par le 68000.



BSR \$123456
JSR 24(a3,a2.l)

[I]nterrupt Priority Level

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

S'il est supérieur à 5, Adébog utilise alors sa propre gestion clavier. Ce mode est très utile pour tracer une routine en **interruption**, faire de la programmation système (timers ...),...

Les accès disque dur sont alors impossibles (le système n'y étant pas adapté). Néanmoins, les accès disquette sont toujours possibles.

 **[I] IPL interne: 7**

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

 **[I] IPL interne: 3**

Repasse en IPL 3 (état "habituel").

[H]istory

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 dans la liste d'une ligne.

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

[Sft_Flèche bas] ou [Espace]: descendre dans la liste d'une page.

[Sft_Flèche haut] ou [Sft_Espace]: remonter dans la liste d'une page.

[Home]: amène à la dernière instruction tracée.

[P]: impression de la liste.

[S]: sauvegarde de la liste.

[Alt_M]emory free

Afficher la **mémoire système libre** en nombre d'octets.

[F1]

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

[F2]

Echanger l'adresse de la **marque** avec l'adresse de la **fenêtre courante**.

[Alt_P]rint

Imprimer le contenu de la **fenêtre courante** (sur toute imprimante matricielle branchée sur le port parallèle

(Centronics)).

[P]rint

Désassembler sur I)mprimante ou D)isque une partie de la mémoire avec possibilité de création d'étiquettes automatique.
Format: source, longueur, adresse d'une zone mémoire inutilisée pour la création des étiquettes, nom du fichier.



[P]

Désassembler <début, longueur>:	{4}.10
Adresse des étiquettes <adresse>:	e0000
Sortie sur D)isque I)mprimante	D
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 \$e0000 sera utilisée en tant que tampon intermédiaire pour la génération automatique des étiquettes (labels).

[E]value

Evaluer une expression.

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



[E] **Evalue:** 2+2

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

[Alt_@]

Message d'information générale.

[Ctl_Alt_Del]

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

 Raccourci clavier identique en standard à partir du TOS 1.4.

[Sft_Ctl_Alt_Del]

Reset à froid (effacement de toute la mémoire).

 Raccourci clavier identique en standard à partir du TOS 1.4.

[Sft_Alt_Help]

Interrompt le programme débogué en cours d'exécution.

 Cette fonction utilisant une routine placée en liste VBL ("VblQueue"), il faut que la Vbl du système soit appelée (IPL du programme en cours de débogage inférieur à 4).

[O]utput

R)S232 E)cran

Passer l'affichage des fenêtres d'Adébog 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 connexions 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 de la RS232** dans les **préférences**.

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

APPENDICES

Appendice 1 - 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'Adébog.

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. Cet emplacement se nomme l'**adresse** et transite par le bus du même nom. Ainsi un microprocesseur comme le 68000 qui équipe le ST et qui peut **adresser** 16 megaoctets a un **bus d'adresse** de 24 bits (puisque'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 68000, 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 dits **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 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'Adébog 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 68000, 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 façon différente en fonction du résultat d'un calcul.

Agir de façons différentes 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 des Codes 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 "Zero"). 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 "Negative")... 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 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 sous-programme (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 68000) contient l'adresse de la mémoire dans laquelle le microprocesseur aura stocké l'adresse qui devra est mise dans le PC à la fin du sous-programme courant. Si un nouveau sous-programme est appelé avant la fin du sous-programme 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 réservée pour stocker toutes ces adresses. Quand le 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 **timer** qui permet au microprocesseur d'exécuter certaines tâches répétitives comme, par exemple, la du clavier et de la souris pour voir si l'un ou l'autre a été actionné et pour agir en conséquence. Dans l'Atari, on compte au moins 4 timers de ce genre.

Enfin, sachez que le 68000 peut travailler dans 2 modes. Le mode **utilisateur** qui interdit au programme en cours d'accéder à certaines zones importantes de la mémoire, en écriture et/ou en lecture, 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. Les routines de la ROM aussi.

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**.

On dispose avec le 68000 de 12 modes d'adressage.

adressage immédiat

L'opérande est compris dans le corps du programme. Le processeur n'a pas besoin d'en connaître l'adresse puisque la donnée fait partie de l'instruction.

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

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.

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 post-incrémentation

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 pré-décrémentation

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).

 `MOVE.B D0,$442(A1)` stocke les 8 bits de poids faible de D0 à l'adresse pointée par A1 plus \$442.

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 bits ou sur 32 au choix du programmeur.

 `MOVE.W $442(A1,D5,L),D0` stocke dans le registre D0 le mot contenu à l'adresse pointée par A1 plus #\$442 plus les 16 bits de poids faible du registre D5.

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 où 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 $442(PC,D0.L),D1` stocke dans D1 l'octet qui se trouve en $PC + 2 + 442 + D0$.

 Ces deux derniers modes d'adressage permettent la génération d'un programme relogable.

Les instructions du 68000.

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**.

Les instructions de traitement arithmétique

ADD SUB CLR NEG NOT DIVU MULU ASR/L EXT SWAP

Addition (+), soustraction (-), mise à zéro, négation (!), inversion, division (/) (résultat sur un mot), multiplication (*) (opérande et résultat sur un mot), décalage arithmétique vers la droite/gauche, extension de signe, échange du mot bas et du mot haut d'un long mot.

Les instructions de traitement logique

AND EOR OR LSR/L ROR/L ROXL/R

Et (&), ou exclusif (~), ou inclusif (!), décalage logique à droite/gauche, rotation vers la droite/gauche, rotation vers la droite/gauche en conservant la retenue.

Les instructions de manipulation de bits

BCHG BCLR BSET BTST

Inverse l'état d'un bit, met à zéro un bit, met à un un bit, teste l'état d'un bit.

Les instructions d'entrées/sorties

MOVE MOVEM LEA PEA EXG MOVEP

Met une valeur, met plusieurs valeurs, met une adresse dans un registre d'adresse, empile une adresse, échange deux valeurs de registres, met une valeur en prenant un octet sur deux.

Les instructions de test

CMP TST TAS Scc Bcc DBcc

Compare, teste, teste puis met à -1, met à 1 si condition vraie (à 0 sinon), branche si condition vraie, teste le registre et branche si condition vraie.

Les instructions de branchement

BRA JMP BSR JSR TRAP RTS RTE

Branche (à moins de 32 Ko), branche (sans limites), branche avec empilement (retour par RTS) limité à 32 Ko, branche avec empilement (retour par RTS), branche par une exception TRAP (retour par RTE), revient d'un BSR ou JSR, revient d'une interruption (voir chapitre sur les interruptions).

Les instructions diverses

LINK NOP

Sauve la valeur du registre spécifié dans la pile, et lui donne la valeur courante de la pile. 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 se 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 s'est passé), il suffit d'appuyer sur [Ctl_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 68000 possède un **Status Register (SR)** dont presque chaque bit possède une signification.

Le SR courant est affiché dans la fenêtre 1 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 68000 de générer une exception "trace" après chaque instruction exécutée.

⊗ Le mode **Trace** correspondant à un vecteur d'exception du 68000, 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 Adébog 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 68000. 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 serait 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 encore 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 lecture pour aller prendre l'appel, discuter quelque temps avec votre interlocuteur, puis reprendre votre lecture.

Cet exemple reflète assez fidèlement le mécanisme de fonctionnement de ce que l'on nomme **interruption** au niveau d'un processeur tel que le **MC68000**. 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 faire autre chose 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 l'ordinateur.

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 68000

Au niveau du 68000, les interruptions sont gérées comme des sous-programmes.

Si, lors de l'exécution d'un programme principal, une interruption survient, le 68000 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 principal 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 68000 si l'interruption correspondante survient.

N° de Vecteur	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	
7	\$1C	Instruction TRAPV	
8	\$20	Violation de Privilège	
9	\$24	Trace	
\$A	\$28	Ligne A	
\$B	\$2C	Ligne F	
\$C-\$E	\$30-\$38	Réservé	
\$F	\$3C	Interruption Non Initialisée	
\$10-\$17	\$40-\$5C	Réservé	
\$18	\$60	Interruption Parasite	
\$19	\$64	Autovecteur (IPL1)	
\$1A	\$68	Autovecteur (IPL2)	Interruption HBL sur l'ATARI
\$1B	\$6C	Autovecteur (IPL3)	
\$1C	\$70	Autovecteur (IPL4)	Interruption VBL sur l'ATARI
\$1D	\$74	Autovecteur (IPL5)	
\$1E	\$78	Autovecteur (IPL6)	Interruption MFP sur l'ATARI
\$1F	\$7C	Autovecteur (IPL7)	
\$20-\$2F	\$80-\$BC	Trap 0-F	

Figure 10: Vecteurs d'interruption

Convention

Certaines interruptions sont provoquées par des instructions du 68000 (CHK, ILLEGAL, TRAP ...) et sont alors appelées exceptions.

Prise en charge d'une interruption par le 68000

Lorsque le 68000 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. Le PC (Compteur ordinal) est empilé.
5. Le SR est empilé.
6. Pour les exceptions **Erreur de bus** et **Adresse impaire**, le processeur empile également:
 - 6A. Un mot contenant l'instruction ayant provoqué l'erreur.
 - 6B. Un mot long contenant l'adresse véhiculée sur le bus lors de l'erreur.
 - 6C. Un mot dont seuls les 5 bits les moins significatifs sont utilisés et ont la signification suivante:
 - 6Ca. Bit 0-2 signaux FCO-2.
 - 6Cb. Bit 3 à 1 signifie que l'erreur est survenue en **espace CPU**.
 - 6Cc. 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**.
7. 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 le SR et le PC.

Priorité des interruptions

Si, lors de l'exécution d'une instruction, plusieurs interruptions surviennent, le 68000 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.

☛ Les trois principales exceptions rencontrées sont:

Erreur de bus: le programme essaie d'accéder en mode utilisateur à des données accessibles uniquement en mode superviseur, ou bien l'adresse pointée n'est pas lisible et/ou inscriptible.

Erreur d'adresse impaire: le programme essaie d'accéder à un mot ou un long mot situé à une adresse impaire.

Instruction illégale: l'instruction n'est pas reconnue par le 68000. Souvent dû à une erreur de pile ou à un écrasement de la mémoire.

Appendice 2 - Périphériques utilisables

Ecran

Adébog gère un écran et une résolution **physiques** (les siens) et **logiques** (ceux du programme en cours de débogage). Le passage de l'un à l'autre 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' Adébog sont celles du système d'exploitation. En conséquence l'accès au disque dur ne peut s'opérer en IPL supérieur à 5. Le message

Pas d'opérations disque dur en IPL>5.

apparaîtra en cas d'erreur.

En revanche les noms de fichier sont expansés, i.e. les caractères joker standards du système d'exploitation (? *) sont pris en compte.

Du fait qu'il n'y a pas de souris sous Adébog et qu'il utilise son propre écran physique, il lui est impossible d'utiliser les boîtes d'alerte d'erreur disque standard. En conséquence, en cas d'erreur disque, Adébog renvoie simplement le type d'erreur après avoir désélectionné le lecteur si nécessaire.

La sortie RS232 (port RS232 - V24)

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

Par défaut, Adébog est configuré pour l'utilisation d'un minitel. Si c'est le cas, Adébog 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 Videotex (40 colonnes). Tout le reste est automatique et Adébog 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 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 de la RS232:

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

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

Figure 11:

Codage de la vitesse de la RS232

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 de la RS232:

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 0	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 12:

Codage de la parité de la RS232

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).

L'imprimante (port parallèle - Centronics)

Afin de pouvoir utiliser l'imprimante même en **IPL 7**, Adébog utilise ses propres routines d'impression par le **port parallèle**. De ce fait il est inutile de reconfigurer (par exemple avec 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 connexions et recommencer.

Appendice 3 - L'évaluateur

Plus que tout autre **débogueur**, Adébog 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 68000** ainsi que les tests et les affectations.

Formats de constantes acceptés

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

 \$53 = %01010011 = \83 = @123

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

 \$53+\10-@45

 Toute constante sans préfixe est considérée en notation hexadécimale.

Opérateurs arithmétiques

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



$$10-(8/2) \quad (=6)$$

$$(10-8)/2 \quad (=1)$$

$$10-8/2 \quad (=6)$$

car la division a priorité sur l'addition.

Opérateurs logiques

Tous les opérateurs logiques disponibles dans le jeu d'instructions du 68000 sont reconnus:

Nom	Instruction 68000	préfixe
OR exclusif	EOR	^
OR inclusif	OR	
ET	AND	&
INVERSE	NOT	~
Décalage logique à gauche	LSL	<<
Décalage logique à droite	LSR	>>

Figure 13

Opérateurs logiques de l'évaluateur



$$55|aa \quad (=\$ff)$$

$$55\&aa \quad (=0)$$

$$1<<6 \quad (=\$40)$$

$$80>>3 \quad (=10)$$

Points d'arrêt

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



#2+10 (=adresse du point d'arrêt n° 2 + \$10)
 {#1}.w=4e71 (place l'instruction NOP à la place du point
 d'arrêt n° 1)

Fenêtres

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



w2=w3+20 (la fenêtre 2 pointe sur la fenêtre 3+\$20)

Indirections et parenthèses

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



((-2+(5*4)+4)-5)+8

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).



{44e}

adresse vidéo logique

{{ff8201}.b<<8+{ff8203}.b)<<8

adresse vidéo physique

☞ 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 et routines (LA, LR, 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 le disque original d'Adébog).

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**, **EX**, **BL** ou **RO** renverra 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 crochets ([]).

 `efface_bloc[{44e},\32000]`

Les paramètres {44e} (adresse de début d'écran logique) et \32000 sont passés à la routine `efface_bloc`

 Une routine peut recevoir au maximum 50 paramètres.

Registres du 68000

Tous les registres du 68000 sont utilisables pour les calculs:

D0 à D7, **A0 à A6**, **A7** (pile utilisateur), **SSP** (Pile Superviseur), **SR** (Registre d'Etat), **CCR** (Registre des Codes de Condition), **PC** (Compteur Ordinal).


d0+a3
ssp-8
ccr&fe

 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.

Tests et affectations

Tests supportés:

`==` (égal à)

`>=` (supérieur ou égal à)

`<=` (inférieur ou égal à)

`>` (supérieur à)

`<` (inférieur à)

`!=` (différent de)

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



`(a0==78000)|(d7==3)` (-1 si `a0==78000` ou si `d7==3`, sinon 0)

`(d0!=-8)` (-1 si `d0` différent de -8, sinon 0)

Affectations:

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

`registre=chaîne` (affectation d'un registre)



`(d0=-1)+(ccr=fe)`

`[adresse]=chaîne` (affectation en mémoire)



`{44e}={{f8201}.b}<<8+{{f8203}.b}<<8`

`variable=chaîne`



`loto=d0+4`

Format des données

L'évaluateur accepte les suffixes **.b** (octet), **.w** (mot) et **.l** (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.

Récurtivité 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)

<<	>>	~		^	&	/	*	-	+	==	!=	>=	<=	<	>	=
----	----	---	--	---	---	---	---	---	---	----	----	----	----	---	---	---

Figure 14:

Priorité des opérateurs de l'évaluateur.

Appendice 4 - Version cartouche. Changement de moniteur en cours de session

La version **cartouche** d'Adébog et l'**interface de changement de moniteur** sont disponibles auprès d'Arobace

Appendice 5 - Index thématique des messages d'erreur

b signifie octet.

w signifie mot.

l signifie long mot.

s signifie chaîne de caractères.

d signifie décimal.

b signifie hexadécimal.

Option non autorisée.

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

Plus de place en vblqueue.

Impossible d'installer l'interruption par [Sft_Alt_Help].

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 Adébog. Elle peut être générée à la suite d'un écrasement partiel d'Adébog.

Imprimante non prête.

Vérifiez les connexions 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.

Pas d'opération disque dur autorisée en IPL>5.

Quand l'IPL interne d'Adébog est supérieur à 5, le système n'autorise plus les accès disque dur.

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 dans le répertoire.

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 [Ctl_L] ne correspond pas à un fichier relogable.

Erreur de relocation.

Le programme chargé avec la fonction [Ctl_L] n'a pas une table de relocation correcte.

TOS inconnu PTerm arrêtera le débogueur.

Cette version du système d'exploitation étant inconnue, l'appel système de terminaison de programme arrêtera aussi Adébug.

Type de symboles inconnu.

Le programme chargé avec la fonction [Ctl_L] possède des symboles de type inconnu.

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 Adébog.

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 à l'initialisation des variables par 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 7 - Table ASCII

Insertion de la table ASCII

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	
0		0		0	e	P	`	p	ç	É	á	ã	ij	0	α	≡
1	0	!	!	1	A	O	a	q	ü	æ	í	õ	ij	B	β	±
2	0	2	"	2	B	R	b	r	é	Æ	ó	ß	K	9	Γ	≥
3	0	3	#	3	C	S	c	s	â	ô	ú	ß	1	Y	ν	≤
4	0	4	\$	4	D	T	d	t	ÿ	ö	ñ	e	λ	ρ	Σ	∫
5	0	5	%	5	E	U	e	u	à	ò	ñ	E	T	Γ	σ	J
6	0	6	&	6	F	V	f	v	ã	û	ß	À	ñ	0	μ	÷
7	0	7	'	7	G	W	g	w	ç	ù	0	À	1	π	τ	≈
8	✓	B	(8	H	X	h	x	ê	ÿ	¿	ö	T	1	ÿ	°
9	0	9)	9	I	Y	i	y	ë	ö	„	„	π	Γ	0	°
A	+	@	*	:	J	Z	j	z	è	ü	„	„	U	0	Ω	•
B	„	£	+	;	K	[k	{	ï	ç	½	„	„	η	δ	√
C	£	£	,	<	L	\	l		î	£	½	„	„	γ	φ	^
D	£	„	-	=	M]	m	}	ì	¥	1	0	„	§	φ	²
E	„	„	.	>	N	^	n	~	ñ	β	«	0	„	^	ε	³
F	„	„	/	?	O	_	o	Δ	ñ	f	»	™	„	∞	π	ˉ

Appendice 8 - Petite documentation de l'Atari ST, STE et TT

Adr.	Taille	Désignation	Explication
------	--------	-------------	-------------

CARTOUCHE			
\$1e0000		cartridge	Adresse de base de la ROM des cartouches.

Table 15: Cartouche.

MEMOIRE			
\$118001	B	memconf	Registre de configuration mémoire.

Table 16: Mémoire.

AFFICHAGE			
\$118201	B	dbaseh	Octet haut de l'adresse de la mémoire vidéo.
\$118203	B	dbasem	Octet médian de l'adresse de la mémoire vidéo.
\$11820d	B	dbasel	Octet bas de l'adresse de la mémoire vidéo (à partir du STE seulement).
\$118205	B	vcounthl	Octet haut du compteur d'adresse vidéo (en écriture à partir du STE seulement).
		{00----}	
\$118207	B	vcounthd	Octet médian du compteur d'adresse vidéo (en écriture à partir du STE seulement).
		{----}	
\$118209	B	vcounlll	Octet bas du compteur d'adresse vidéo (en écriture à partir du STE seulement).
		{----0}	
\$11820a	B	syncmode	
		{----00}	
			0: Synchro Interne / 1: Externe.
			0: Fréquence 50 Hz / 1: 60 Hz.
\$118240	16W	Palette de couleur.	
		{----000}	Sur les ST et STF.
		{---0000}	Sur le STE.
\$118260	B	shiftmode	Résolution vidéo:
			0: basse résolution 320*200 4 plans.
			1: moyenne résolution 640*200 2 plans.
			2: haute résolution 640*400 1 plan.

Table 17: Affichage.

DMA			
\$f18804	B/W/	diskctl	Registre de contrôle.
\$f18806	B/W	fifo	Registre de données.
\$f18809	B	dmahigh	Octet haut de l'adresse du pointeur DMA.
\$f1880b	B	dmamed	Octet médian de l'adresse du pointeur DMA.
\$f1880d	B	dmalow	Octet bas de l'adresse du pointeur DMA.
\$f18800	B/W/		
		En lecture:	
		g1read	Lecture du port B du processeur sonore.
		En écriture:	
		g1select	Sélecteur de registre du processeur sonore.
\$f18802	B/W		
		En écriture:	
		g1write	Ecriture sur les ports A ou B du processeur sonore.
		(00000000)	
			Sélection de face de disque.
			Sélection du disque A.
			Sélection du disque B.
			Signal RTS de la RS232.
			Signal CTS de la RS232.
			Signal STROBE de la Centronic.
			Sortie à usage général.
			Réservé.

Table 18: DMA.

MFP			
\$ffa00	B	mfp	Adresse de base du MFP 68901.
\$ffa01	B	gpiop	Entrée/Sortie à usage Général.
\$ffa03	B	aer	Sélection du front (montant ou descendant) actif pour chaque bit de la ligne GPIOP.
\$ffa05	B	ddr	Sens de circulation des données sur la ligne GPIOP.
\$ffa07	B	iera	Activation d'interruptions.
		(00000000)	
			Timer B.
			Erreur d'émission.
			Tampon d'émission vide.
			Erreur de réception.
			Tampon de réception plein.
			Timer A.
			Détection de sonnerie de la RS232.
			Détection du Moniteur monochrome.
\$ffa09	B	ierb	Activation d'interruptions.
		(00000000)	
			Signal 'BUSY' de la Centronic.
			Signal RTS de la RS232.
			Signal CTS de la RS232.
			Inutilisé.
			Timer D.
			Timer C.
			ACIA Clavier et ACIA Midi.
			DMA.
\$ffa0b	B	lpra	Drapeau de mise en attente d'interruptions (même structure de bits que iera).
\$ffa0d	B	lprb	Drapeau de mise en attente d'interruptions (même structure de bits que ierb).
\$ffa0f	B	lsra	Mise en service d'interruptions (même structure de bits que iera).
\$ffa11	B	lsrb	Mise en service d'interruptions (même structure de bits que ierb).
\$ffa13	B	lmra	Masque d'interruptions (même structure de bits que iera).
\$ffa15	B	l mrb	Masque d'interruptions (même structure de bits que ierb).
\$ffa17	B	vr	
		(00000000)	
			Drapeau d'interruption automatique.
			Bit 4 du n° de vecteur d'interruption.
			Bit 5 du n° de vecteur d'interruption.
			Bit 6 du n° de vecteur d'interruption.
			Bit 7 du n° de vecteur d'interruption.
			(Les bits 0 à 3 sont générés par le MFP).
\$ffa19	B	tacr	Registre de contrôle du timer A.
\$ffa1b	B	tbcr	Registre de contrôle du timer B.
\$ffa1d	B	tcdr	Registre de contrôle des timers C et D.
\$ffa1f	B	tadr	Registre de décompte du timer A.
\$ffa21	B	t bdr	Registre de décompte du timer B.
\$ffa23	B	tcdr	Registre de décompte du timer C.
\$ffa25	B	tdcr	Registre de décompte du timer D.
\$ffa27	B	ucr	Registre de contrôle de l'USART.

\$ffa2b	B	rsr	Registre d'état du récepteur.
		(00000000)	
			Validation du récepteur.
			Validation du caractère de synchronisation.
			Réception du caractère de synchronisation.
			Erreur de format.
			Erreur de parité.
			Surcharge.
			Tampon plein.
\$ffa2d	B	tsr	Registre d'état du transmetteur.
		(00000000)	
			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.
\$ffa2f	B	udr	Registre de données de l'USART.

Table 19: MFP.

CLAVIER			
\$fff00	B	keyctl	
		En écriture:	Registre de contrôle de l'ACIA clavier.
		[00000000]	
			Vitesse de transmission.
			Protocole série.
			Type de validation d'interruption.
			Drapeau interruptions permises.
		En lecture:	Registre d'état de l'ACIA clavier.
		{00000000}	
			Tampon plein.
			Données disponibles.
			Signal DCD.
			Signal CTS.
			Erreur de transmission.
			Surcharge.
			Erreur de parité.
			Drapeau d'interruption.
\$fff02	B	keybd	Registre de données de l'ACIA clavier.
\$fff04	B	midictl	Registre d'état et de contrôle de l'ACIA midi. (Identique à celui de l'ACIA clavier (fff00)).
\$fff06	B	midid	Registre de données de l'ACIA midi.

Table 20: Clavier.

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	B	proc_enum	Numéro d'exception.
\$3c8	L	proc_esp	Valeur de la pile utilisateur lors de l'effondrement.
\$3cc	L	proc_stk	16 mots du dessus de la pile superviseur.

Table 21: Informations Post-Mortem.

PAGE 4			
\$400	L	etv_tlmar	Vecteur timer C de maintenance du multilâche du GEM.
\$404	L	etv_critlc	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 pointé sur RTS).
\$40c	L	etv_xlra	Espace pour 5 vecteurs réservés pour extensions futures.
\$420	L	memvalid	Drapeau indiquant la validité des données mémoire 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	B	memconf	Contient une copie du registre de configuration d'adresse memconf (\$fff8001).
\$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	reavector	Contient l'adresse de la routine reset exécutée si \$426 contient \$31415926.
\$42e	L	physlop	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 (CI \$420). Sa valeur normale est \$E3789BAA.
\$43e	B	lock	Drapeau qui bloquant la routine Vbl de désélection de disque quand sa valeur est non nulle.
\$440	W	seekrate	Vitesse de déplacement piste à piste de la tête de lecture du drive utilisées lors des appels disque de la ROM. 0: 6 ms. 1: 12 ms. 2: 2 ms (à ne pas utiliser sur ST). 3: 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	_lverfly	Drapeau de vérification des accès disques en écriture (pour RWABS (BIOS 4)).
\$446	W	_bootdev	Numéro du drive à partir duquel le système a 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	delshllmd	Résolution utilisée lors de la commutation de monochrome en couleur.
\$44c	B	sshiftmd	Résolution graphique de l'application courante. Il s'agit de la copie du registre vidéo d'adresse \$fff8260.
\$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	irclock	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 connexion ou de la déconnexion 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_medtach	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	B	conterm	Drapeaux permettant de configurer le clavier:
		(---0000)	
			Bruit lors de la pression des touches (géré par le timer C).
			Répétition des touches.
			Bruit par pression sur [Ctrl G].
		└───	CONIN (BIOS 2) délivre l'état des touches Alt, Ctrl, Shift 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	ihemd	Pointeur sur le début d'une liste chaînée 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	savplr	Pointeur sur un espace de sauvegarde temporaire des registres lors d'un appel BIOS.
\$4a6	W	nflops	Nombre de lecteurs de disquette connectés. 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).
\$4ae	L	sav_context	Pointeur sur un espace de sauvegarde des registres utilisés lors d'un traitement d'exception.
\$4b2	L	_buf1	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	Chaîne d'environnement par défaut, (4 octets nuls).
\$4c2	L	_drvblks	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 un header du début du TOS.
\$4f6	L	shell_p	Pointeur sur l'adresse du Shell.
\$4fa	L	end_os	Pointeur sur la fin de la zone réservée au TOS en RAM (Le 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.

\$502	L	scr_dump	Pointeur sur une routine appelée si _dump1g (\$40e) 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_isto	Pointeur sur une routine déterminant l'état du port Centronics. Utilisé par la routine de copie d'écran.
\$50a	L	prv_ist	Pointeur sur une routine de sortie Centronics. Utilisé par la routine de copie d'écran.
\$50e	L	prv_auxo	Pointeur sur une routine déterminant l'état du port RS232. Utilisé par la routine de copie d'écran.
\$512	L	prv_aux	Pointeur sur une routine de sortie RS232. Utilisé par la routine de copie d'écran.
\$516	L	pun_ptr	Pointeur sur une structure si un disque dur est connecté et AHDI chargé.
\$51a	L	memval3	Drapeau de validité de la mémoire (Cf 420).
			Les 32 pointeurs suivants sont disponibles à partir du TOS 1.2.
\$51e	8L	bconstat_vec	8 pointeurs sur la routine bconstat pour les divers périphériques (BIOS 1).
\$53e	8L	bconin_vec	8 pointeurs sur la routine bconin pour les divers périphériques (BIOS 2).
\$55e	8L	bcostat_vec	8 pointeurs sur la routine bcostat pour les divers périphériques (BIOS 8).
\$57e	8L	bconout_vec	8 pointeurs sur la routine bconout pour les divers périphériques (BIOS 3).
\$59e	B	proc_type	Drapeau 88030 (Sur TT uniquement).
\$5ac	L	prv_clk	Pointeur sur la routine produisant le bruit des touches du clavier (Sur TT uniquement).
\$e2e	L	mouse_vec	Pointeur de la gestion souris appelé en liste Vbl.
\$e4a	L	timer1	
\$e4e	L	timer2	
\$ea4	W	timerc_div	Registre de décalage du timer C permettant la division de la période d'horloge par 4.

Table 22: Page 4.

\$5a0 → pointeur sur cookie jar

BLITTER			
\$118a00		bltbase	Adresse de base des registres du blitter.
\$118a00 \$118afe	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.
\$118a20	W	src_xinc	Nombre d'octets qui séparent les adresses de 2 mots à copier consécutifs d'une même ligne graphique.
\$118a22	W	src_yinc	Nombre d'octets qui séparent 2 lignes consécutives du bloc graphique à copier dans une page écran.
\$118a24	L	src_addr	Adresse du premier mot du bloc à copier.
\$118a28	W	endmask1	Masque du premier mot d'une ligne du bloc à copier.
\$118a2a	W	endmask2	Masque d'un mot du milieu (i.e. ni le premier mot, ni le dernier) d'une ligne du bloc à copier.
\$118a2c	W	endmask3	Masque du dernier mot d'une ligne du bloc à copier.
\$118a2e	W	dst_xinc	Même fonction que src_xinc (\$118a20) mais pour le bloc destination.
\$118a30	W	dst_yinc	Même fonction que src_yinc (\$118a22) mais pour le bloc destination.
\$118a32	L	dst_addr	Même fonction que src_addr (\$118a24) mais pour le bloc destination.
\$118a36	W	x_count	Nombre de mots par ligne du bloc destination.
\$118a38	W	y_count	Nombre de lignes du bloc destination.
\$118a3a	B	HOP	Opération logique entre la trame de demi-teinte, définie par les registres de demi-teinte 0 à 15 (\$118a00 à \$118afe), et le bloc source. 0: (source) OU (trame). 1: trame. 2: source. 3: (source) ET (trame).
\$118a3b	B	OP	Opération logique entre le bloc source et le bloc de destination.
		{000-0000}	
			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 en fin d'exécution.

\$118a3d	B	{00--0000}	
			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.

Table 23: Bitter

		GOPROCESSEUR ARITHMETIQUE 68881 (FPU)	
\$111a40	W	FPCR	Registre status.
\$111a42	W	FPCTL	Registre contrôle.
\$111a44	W	FPSAVE	Registre de sauvegarde (inutilisé dans la librairie ALCYON).
\$111a46	W	FPREST	Registre de restauration (inutilisé dans la librairie ALCYON).
\$111a48	W	FPOPR	Registre de mot d'opération.
\$111a4a	W	FPCMD	Registre de commande.
\$111a4c	W	FPRES	Réservé.
\$111a4E	W	FPCCR	Registre de code condition.
\$111e50	L	FPOP	Registre d'opérande.

Table 24: Coprocesseur mathématique 68881 (FPU)

Déc	Hexa		LIGNE A	
-906	-38a	L	CUR_FONT	Adresse du descripteur de la fonte courante.
-856	-35B	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	Modé d'écriture du curseur de la souris:
				1: normal.
				-1: 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	INO_TAB	Résultat de l'appel VDI vq_extend.
-692	-2b4	45 W	DEV_TAB	Résultat de l'appel VDI vq_extend.
-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	Etat courant du bouton de la souris.
				Bk 0: bouton gauche.
				Bk 1: 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 ligne et 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	B	CUR MS STAT	Etat courant de la souris: Bit 0: bouton gauche. Bit 1: bouton droit. Bit 2~4: réservés. Bit 5: souris déplacée. Bit 6: modification de l'état du bouton droit. Bit 7: 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	B	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	Etat de la sauvegarde des informations souris. Bit 0: Validité du contenu du tampon de sauvegarde. Bit 1: Largeur des données sauvegardées (0:Mot/1:Long mot).
-322	-142	128W	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_OFF SET	Nombre d'octets par ligne de texte (nombre d'octets par ligne * 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	B	CURSOR_FLASH	Fréquence de clignotement du curseur de texte (en nombre de VBL).
-23	-17	B	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.
-18	-10	W	FIRST_ASCII	Code ASCII du premier caractère de la police courante.
-14	-e	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	B	ALPHA_STATUS	Drapeaux du mode d'affichage.
-4	-4	W	MAX_Y	Hauteur de l'écran en pixel.
-2	-2	W	BYTES_LN	Nombre de caractères 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.
+6	+6	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.
+28	+1c	W	COLBIT2	Valeur du plan 2 de la couleur.
+30	+1e	W	COLBIT3	Valeur de plan 3 de la couleur.
+32	+20	W	LSTLIN	Vaut -1 (\$FFFF).
+34	+22	W	LNMASK	Style de ligne.
+36	+24	W	WMODE	Mode d'écriture (0 à 3).
+38	+26	W	X1	Abscisse du point n°1.
+40	+28	W	Y1	Ordonnée du point n°1.
+42	+2a	W	X2	Abscisse du point n°2.
+44	+2c	W	Y2	Ordonnée du point n°2.
+46	+2e	L	PATPTR	Adresse du motif de remplissage.
+50	+32	W	PATMSK	Masque de remplissage.
+52	+34	W	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 ou 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	Épaisseur 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	SCRTPHP	Adresse d'un tampon pour les effets spéciaux.
+112	+72	W	SCRPT2	Offset du tampon d'agrandissement dans SCRTPHP.
+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.

Table 25: Ligne A

JOYSTICK (STE)			
\$119200	L	joy_fire	Appui du bouton fire.
		{-----}{----0000}	
\$119202	L	joy_pos	Positions.
			3 1 2 0
			{HBGDHBGDHBGDHBGD} (H:haut B:bas G:gauche D:droite)
\$119210	L	joy0_x	Position x du joystick analogique 0.
		{-----}{00000000}	
\$119212	L	joy0_y	Position y du joystick analogique 0.
		{-----}{00000000}	
\$119214	L	joy1_x	Position x du joystick analogique 1.
		{-----}{00000000}	
\$119216	L	joy1_y	Position y du joystick analogique 1.
		{-----}{00000000}	
\$119220	L	joy2_x	Position x du stylo/pistolet optique.
		{-----00}{00000000}	
\$119222	L	joy2_y	Position y du stylo/pistolet optique.
		{-----00}{00000000}	

Table 26: Joystick STE

AFFICHAGE (STE)			
\$1f820d	B	vbase0	Octet bas de l'adresse de l'écran physique.
		{-----0}	
\$1f8265	B	hscrofl	N° du bit de départ de l'adresse physique de l'écran (0-15).
		{0000----}	
\$1f8201	L	lnewid	Longueur logique d'une ligne de l'écran.
		{00000000}{-----}	

Table 27: Affichage STE.

SON DMA (STE)			
\$1f8901	B	dma_sound_enable	Registre de contrôle du son. 00: interdit le son. 01: joue le son une seule fois. 10: réservé. 11: joue le son en boucle.
		{-----00}	
\$1f8902	L	fbasehl	Octet haut de l'adresse de base du son.
		{00---}{-----}	
\$1f8904	L	fbasemid	Octet médian de l'adresse de base du son.
		{-----}{-----}	
\$1f8906	L	fbase0	Octet bas de l'adresse de base du son.
		{-----}{-----0}	
\$1f8908	L	cbasehl	Octet haut de l'adresse courante du son.
		{00---}{-----}	
\$1f890a	L	cbasemid	Octet médian de l'adresse courante du son.
		{-----}{-----}	
\$1f890c	L	cbase0	Octet bas de l'adresse courante du son.
		{-----}{-----0}	
\$1f890e	L	ebasehl	Octet haut de l'adresse de fin du son.
		{00-----}{-----}	
\$1f8910	L	ebasemid	Octet médian de l'adresse de fin du son.
		{-----}{-----}	
\$1f8912	L	ebase0	Octet bas de l'adresse de fin du son.
		{-----}{-----0}	
\$1f8921	L	sound_ctrl	Mode et fréquence.
		{m---r}	
			m (mode) 0: stereo. 1: mono. r (fréquence). 00: 6258 Hz. 01: 12517 Hz. 10: 25033 Hz. 11: 50066 Hz.
\$1f8922	L	sound_data	Registre de donnée de l'interface Microwire. (Réglage du volume et de la tonalité).
		{00000000}{00000000}	
\$1f8924	L	sound_mask	Registre de masquage de l'interface Microwire.
		{00000000}{00000000}	

Table 28: DMA STE

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

Table 29: Commandes de l'interface MicroWire

AFFICHAGE (TT)			
\$1f8262	W	ttres	Résolution du TT.
			0: 320*200 4 plans (ST)
			1: 640*200 2 plans (ST)
			2: 640*400 1 plan (ST)
			4: 640*480 4 plans (TT)
			6: 1280*960 1 plan (TT)
			7: 320*480 8 plans (TT)
\$1f8400	256 W	ttcolor	Palette de couleur.

Table 30: Affichage TT.

Liste des figures

Figure 1:

Les cinq fenêtres d'Adébog et les quatre types de visualisation.

Figure 2:

Une fenêtre en mode plein écran ([Alt_Z]oom de fenêtre).

Figure 3:

Le mode quatre fenêtres par élimination de la fenêtre 1.

Figure 4:

Le mode visualisation ASCII du contenu des registres dans la fenêtre 1.

Figure 5:

Le mode édition du SR et des timers dans la fenêtre 1 ([Alt_E]).

Figure 6:

Le mode édition de mémoire en hexadécimal/ASCII ([Alt_E]).

Figure 7:

Liste des tailles courante et maximum des tampons de variables et points d'arrêt.

Figure 8:

Liste des variables ([L]).

Figure 9:

Liste des points d'arrêt ([Help]).

Figure 10:

Vecteurs d'interruption.

Figure 11:

Codage de la vitesse de la RS232.

Figure 12:
Codage de la parité de la RS232.

Figure 13:
Opérateurs logiques de l'évaluateur.

Figure 14:
Priorité des opérateurs de l'évaluateur.

Figure 15:
Cartouche.

Figure 16:
Mémoire.

Figure 17:
Affichage.

Figure 18:
DMA

Figure 19:
MFP.

Figure 20:
Clavier.

Figure 21:
Informations Post-Mortem.

Figure 22:
Page 4.

Figure 23:
Blitter.

Figure 24:
Coprocasseur mathématique 68881 (FPU).

Figure 25:
Ligne A.

Figure 26:
Joystick STE.

Figure 27:
Affichage STE.

Figure 28:
DMA STE.

Figure 29:
Commandes de l'interface MicroWire.

Figure 30:
Affichage TT.

Index général

= 110

\$24 93

(107

) 107

+ 106

• 106

- 106

.app 60

.b 110

.l 110

.prg 60

.tos 60

.itp 60

.w 110

/ 106

 11

 11

 11

 11

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 10

1.4 Connaissances nécessaires 10

2 Description rapide 11

2.1 Introduction 11

2.10 Changement de moniteur en cours de session 20

2.11 Exemple d'utilisation 21

2.2 Adébog, 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 18

- 2.7 Ligne de commande à l'exécution 18
- 2.8 Usages d'un terminal 19
- 2.9 Version cartouche 20
- 3 Références 24
 - 3.1 Liste thématique des fonctions 24
 - 3.10 Opérations de contrôle de flux du programme 67
 - 3.11 Points d'arrêt 72
 - 3.12 Commandes diverses 77
 - 3.2 Liste alphabétique des fonctions 28
 - 3.3 Fenêtres 32
 - 3.4 Gestion et édition mémoire 39
 - 3.5 Macros 43
 - 3.6 Variables 48
 - 3.7 Ecran 59
 - 3.8 Opérations disque 60
 - 3.9 Configurations 63
 - < 110
 - <= 110
 - = 110
 - > 110
 - >= 110
 - ? 43

A

A0 à A6 109

A0-A7. 36

A7 85; 109

A7' 86

Adébog 2

ADEBUG.SAV 18; 63

ADEBUG.VAR 54

adressage absolu court 89

adressage absolu long 90

adressage direct d'un registre d'adresse 87

adressage direct d'un registre de donnée 87

adressage immédiat 87

adressage indirect d'un registre d'adresse 88

adressage indirect d'un registre d'adresse avec déplacement 89

adressage indirect d'un registre d'adresse avec index et déplacement 89

adressage indirect d'un registre d'adresse avec post-incrémentation 88

adressage indirect d'un registre d'adresse avec pré-décrémentation 88

adressage relatif au PC avec déplacement 90

adressage relatif au PC avec index et déplacement 90

adresse 82

adresse de base d'une fenêtre 107

Adresse impaire 99

Adresse impaire ou illisible. 36; 114

AFFICHAGE 119

AFFICHAGE (STE) 134

AFFICHAGE (TT) 135

Alternate 11

alt_ 17; 43

appeler 85

Appendice 1 - Connaissances nécessaires à l'usage d'un débogueur 81

Appendice 2 - Périphériques utilisables 101

Appendice 3 - L'évaluateur 105

Appendice 4 - Version cartouche. Changement de moniteur en cours de session 112

Appendice 5 - Index thématique des messages d'erreur 113

Appendice 7 - Table ASCII 118

Appendice 8 - Petite documentation de l'Atari ST, STE et TT 119

APPENDICES 81

architecture 84

Arobace 112

Arrière 44

ASCII 62

assembleur 87

Attention! Copie incomplète. 41; 113

Attention! Remplissage incomplet. 41; 114

Avant-propos 2

B

backquote 43
basse résolution 59
BL 49; 63
Bl <s> non trouvé. Pressez une touche. 117
BLITTER 128
bloc 61
bus 82
bus d'adresse 82
bus de donnée 82

C

C 39; 84
C)harger 45
Carry 84
cartouche 112; 119
CCR 84; 109; 110
Chemin invalide. 114
CLAVIER 123
Clr 44
compteur ordinal 36; 83; 99; 109
"Condition Code Register" 84
Control 11
COPROCESSEUR ARITHMETIQUE 68881 (FPU) 129
couleur d'encre 59
couleur de fond 59
ctl_ 17; 43
cycle d'écriture 99
cycle de lecture 99

D

D)isque 79
D0 à D7 109
Début 45
Désassembler 79
destination 87
Directive MAC Inconnue. 116
Disque (souple et dur) 101
DMA 120
donnée 82
DRI 60

E

E)nregistrer J)ouer C)harger V)oir une macro. 44
E)nregistrer T)racer A)rrière P)asser C)lr 44
échangeur électronique 20
Ecran 101
écran 101
écran logique 59
écran physique 59
Enregistrer 44
entrées/sorties 91
EPROMs 20
Erreur d'écriture n°<d> dans le fichier <s>. 115
Erreur d'écriture. 115
Erreur d'impression. 104
Erreur de bus 99
Erreur de création. 115
Erreur de lecture n°<d> dans le fichier <s>. 115
Erreur de relocation. 115
Erreur disque. 114
Erreur FATALE de réservation mémoire dans les préférences.
Pressez une touche. 113
Erreur interne n°<bx> Décalage <lx>. 113
Erreur n°<bd>.VAR <ld>.ligne <ld>
<s>. 117

espace CPU 99
évaluateur 105
EX 53
exception 94
exception TRAP 74
exceptions 74; 98
exemple.s 23

F-G-H

faux 110
FC0-2 99
fenêtre 32
fenêtre 1 12
fenêtre 2 12
fenêtre active 12; 32
fenêtre courante 12; 32
Fenêtres 107
fenêtres 3, 4 et 5 12
Fichier <s> non trouvé. 114
fichier binaire 61; 62
fichier relogeable 60
flèche 93
Fonction inconnue. 116
Format des données 110
Gestion d'une interruption par le 68000 97
haute résolution 59
HIS 14; 63
HiSoft Extended Debug 60
Historique 14

I-J-K

Imprimante 79
Impossible de mettre le point d'arrêt. 72; 116
Imprimante non prête. 113
Imprimer 78
Index général 139
Indirection 108
Indirections et parenthèses 107
INFORMATIONS POST-MORTEM 123
instruction 82
instruction de fin de sous-programme 85
instruction illégale 94
instructions d'entrées/sorties 92
instructions de branchement 92
instructions de manipulation de bits 91
instructions de test 92
instructions de traitement arithmétique 91
instructions de traitement logique 91
instructions diverses 92
interface de changement de moniteur 112
interruption 77; 96; 99
interruptions 85
IPL 7 103
IPL interne 77
Jouer **T**racer **A**rrière **P**asser **V**oir **D**ébut 44
Jouer 44
JOYSTICK (STE) 133

L-M

L'imprimante (port parallèle - Centronics) 103

"Last In, First Out" 85

LA 48; 63

La sortie RS232 (port RS232 - V24) 102

Les instructions du 68000 91

LIFO 85

LIGNE A 130

ligne de commande 14; 18

Liste des figures 136

Liste des points d'arrêt 55; 75

LR 48; 63; 111

MAC 45; 63

macro 43

macros 17

marque 78

Mauvaise adresse de tampon. 114

Mauvaise eval en point d'arrêt. 116

Mauvaise eval en Trace Jusqu'à. 115

Mauvaise pile pour le traçage. 116

MC 45

MC68000 96

MEMOIRE 119

mémoire 81

Mémoire illisible. 113

mémoire morte 82

mémoire système libre 78

mémoire vive 81

MFP 121

Minitel Y(es) N(o) 102

modes d'adressage 87

moniteurs 8

moyenne résolution 59

multifenêtrage 12

N-O-P-Q

N 39; 84
notation décimale 105
notation hexadécimale 105
Notions d'interruption / exception 96
Notions de trace 92
opérande 87
Opérateurs arithmétiques 106
Opérateurs logiques 106
Option non autorisée. 113
PAGE 4 124
Parité de la RS232 103
Pas d'opération disque dur autorisée en IPL>5. 114
Pas d'opérations disque dur en IPL>5. 101
Pas de MAC. 116
Pas de registres sauvés. 114
Pas de VAR. 117
Pas un fichier exécutable. 115
Passages de paramètres dans les routines 109
Passer 44
PC 36; 82; 83; 99; 109
PC impair ou illisible. 116
pile 84; 85
pile superviseur 36; 109
pile utilisateur 109
Plus de mémoire. 115
Plus de place en vblqueue. 113
point d'arrêt 72; 94
Point d'arrêt numéro x mis. 72
Point d'arrêt n°x atteint. 94
Point d'arrêt n°x mis. 94
point d'arrêt simple 94
points d'arrêt 74; 107
points d'arrêts 107
port parallèle 103
Priorité 100

Priorité des interruptions 100 ·
Priorité des opérateurs (en ordre décroissant) 111
Prise en charge d'une interruption par le 68000 99
"Program Counter" 36; 82; 83
programme 60; 83
programme exécutable 8; 60
programme principal 85

R

R 45
R)S232 E)cran 80
raccourcis clavier 12
RAM 81
Recherche non définie. 114
Récursivité et limites de l'évaluateur 111
Registre d'Etat 109
Registre des Codes de Condition 84; 109
registres 82; 83
registres d'adresse 83
registres du 68000 77; 109
Remerciements 2
répertoire 60
Reset à chaud 80
Reset à froid 80
résolution 101
revenir 85
RO 49
Ro <s> non trouvée. Pressez une touche. 117
ROM 82
ROSTRUCT.S 49
RTE 99

S.

S 39

sft_ 17; 43

Shift 11

Sommaire 3

SON DMA (STE) 134

sous-programme 84

sous-programmes 84

SP 85

SR 39; 93; 99; 109; 110

SSP 36; 86; 109

Status Register 93

STO 63

superviseur 86

symboles 9

symbolique 17

système d'exploitation 83

T

T 39; 93

tampon 63

Tampon BL plein. Pressez une touche. 117

Tampon LA plein. Pressez une touche. 117

Tampon LR plein. Pressez une touche. 117

Tampon MAC plein. 116

Tampon VAR plein. Pressez une touche. 117

terminal 19; 80; 102

Tests et affectations 110

tests et branchements 91

TEXT 53; 93

timer 86

Timer A 39

Timer B 39

Timer C 39

Timer D 39

TOS inconnu PTerm arrêtera le débogueur. 115

Trace 93
Trace 6)8020 69
Trace I)instruction 68
Trace J)usqu'à 68
Trace R)ien 68
Tracer 44; 67; 92
traitement de l'information 91
traitement numérique 91
Trap 0
traps 94
Type de symboles inconnu. 115

U-V-W-X-Y-Z

utilisateur 86
V)oir 45
VAR 63
variables 10; 15
Variables, blocs et routines (LA, LR, EX, RO, BL) 108
VEC ;
visualisation ASCII 62
Voir 44
vrai 110
w1~w5 107
"[]" 11
V 39
vecteur 97
Vitesse de la RS232 102
Voir écran logique 70
Voir l'écran logique Y)es N)o 69
X 39
Z 39; 84
Zero 84

[] 109
[-1] 94
[Alt_1~5] 34
[Alt_@] 79
[Alt_A]SCII 62
[Alt_B]reakpoint 72
[Alt_D]irectory 60
[Alt_eX]écute 71
[Alt_E]dition de fenêtre 39
[Alt_L]ock 37
[Alt_M]emory free 78
[Alt_N] 42
[Alt_P]rint 78
[Alt_S]épare/Regroupe fenêtres 33
[Alt_T]ype de visualisation 34
[Alt_V]ariable 48
[Alt_Z]oom de fenêtre 32
[A]dresse de fenêtre 37
[Backspace] 14
[B]inary load 61
[Caps Lock] 11
[Ct] flèche bas] 37
[Ct] flèche haut] 37
[Ct]_0~7] 35
[Ct]_8] 36
[Ct]_9] 36
[Ct]_Alt_Del] 80
[Ct]_Alt_I]nverse 59
[Ct]_A]rrêt 68
[Ct]_B]reakpoint 72
[Ct]_C] 67
[Ct]_Del] 14
[Ct]_D]étourne système 74
[Ct]_eX]écute 71
[Ct]_E]xception 74
[Ct]_F]orce 70

[Ctl_J]ump 70
[Ctl_K]ill 74
[Ctl_L]oad program 60
[Ctl_L]ower 14
[Ctl_O]utput 59
[Ctl_P]références 64
[Ctl_R]un 67
[Ctl_S]kip 67
[Ctl_T]race 70
[Ctl_T]widdle 14
[Ctl_U]ntil 70
[Ctl_U]pper 14
[Ctl_Z] 67
[C]opier 41
[Delete] 14
[D]irectory 60
[Enter] 14
[Esc] 14; 42
[E]value 79
[F10] 20; 59
[F1] 78
[F2] 78
[F8] 20; 59
[F9] 20; 59
[Flèche bas] 36
[Flèche droite] 14; 37
[Flèche gauche] 14; 37
[Flèche haut] 36
[F]ill 41
[G]et expression 42
[Help] 55; 75
[H]istory 78
[I]nterrupt Priority Level 77
[J]ump 70
[K]eep 77
[L]iste des variables 57

[M]acro 44
[N]ext 42
[O]utput 80
[P]rint 79
[Return] 14
[R]estore 77
[Sft flèche bas] 37
[Sft flèche haut] 37
[Sft] 14
[Sft_Alt_Help] 80
[Sft_Ctl_0~7] 36
[Sft_Ctl_8] 36
[Sft_Ctl_9] 36
[Sft_Ctl_Alt_Del] 80
[Slave binary] 62
[Tab] 33
[T]race (Jusqu'à, Instruction, Rien, 68020) 68
[U]ntil 70
[V]oir 59
[W]atch 77
 ' 17: 43
 ' manquant dans macro. 116
 | 108
 | 108
 ° 46
 °b <expression> <étiquette> 47
 °c <texte> 46
 °F 46
 °l <étiquette> 47
 °s 46
 °t 46

A-DÉBOG

Le premier débogueur symbolique
professionnel sur ATARI ST

-Fonctionne sur tous les modèles de la gamme ST (520, 1040, Mega, STE)
dans toutes les résolutions.

Convivial:

- Enregistrement et exécution de macro-commandes
- Multifenêtres
- Nombreux raccourcis clavier
- Différenciation écran physique et logique

Puissant:

- Plus de 100 fonctions

A-DÉBOG offre des fonctions inédites jusqu'alors :

- Visualisation de toutes les entrées/sorties
 - Visualisation de la mémoire en temps réel
 - Gestion du co-processeur arithmétique 68881
 - Mode "trace" avec paramétrage total
 - Points d'arrêts ("Breakpoints") paramétrables en nombre, en évaluation et en type
 - Chargement, évaluation et exécution de routines externes
 - Compatibles avec tous les assembleurs, C et Basics (GFA⁺, Omikron⁺..)
 - Possibilité de désigner une adresse, un bloc ou une routine par un label
- Niveau d'interruption interne réglable de 0 à 7

A-DÉBOG offre aussi :

- 4 types de visualisation : désassemblage, "dump" mémoire, ASCII, registres

A-DÉBOG s'adresse aussi bien aux programmeurs débutants ou chevronnés,
qu'aux auteurs de logiciels.

Avec A-DÉBOG descendez au plus profond de votre ST.



arobace éditions