Cenacle-News

Decembre 1993 No 12

Variable Name Table Editor

SPÉCIAL JEUX D'ACTION

Program Helper

DLI MAKER

Debug+

CLUB CENACLE B.P. 49 95110 SANNOIS

LA BIBLIOTHEQUE DU CLUB: LOGICIELS DU DOMAINE PUBLIC XL/XE

REF.	TITRES	NB FACES	PRIX	REF.	TITRES	NB FACES	PRIX
DP101	DOS 4.0 ATARI	1F	20 Frs	DP102	DISK LABELLER	2F	20 Frs
DP103	FOUR TRACKS	2F	35 Frs	DP104	ARCHIVER MACHINE	1F	25 Frs
DP105	GRAPHMASTER	1F	25 Frs	DP106	SCREEN MAKER	1F	25 Frs
DP107	DECOMPACTEUR XL ou XE	2F	20 Frs	DP108	MIDI TRANSMIT	2F	20 Frs
DP109	SPECIAL DEMOS PAGE 6	4F	30 Frs	DP110	MICROCHECK 3.03	4F	30 Frs
DP111	COMPTABILITE 1029	1F	40 Frs	DP112	COMPTABILITE OKI	1F	40 Frs
DP113	MANDLEBROT SETS	2F	25 Frs	DP114	DISK EDITOR-DISAS	2F	30 Frs
DP115	GRAPHIQUE MACHINE	2F	25 Frs	DP116	MENU MAKERS	2F	30 Frs
DP117	COMPIL No 1 JEUX	2F	25 Frs	DP118	BASIC ROUTINES	2F	25 Frs
DP119	COPYMATE/MyCopyR! 2.1	2F	20 Frs	DP120	TRIVIA QUIZ	2F	25 Frs
DP121	ACE C + Doc français	2F	35 Frs	DP122	PRINT SHOP COLLEC.	6F	75 Frs
DP123	DAISY-DOT II	3F	30 Frs	DP124	ICONES PRINT SHOP	3F	30 Frs
DP125	SIGNMAKERS/TITLESCREEN	2F	25 Frs	DP126	TURBO BASIC + Doc	2F	30 Frs
DP127	TURBO SUPPORT 1	2F	25 Frs	DP128	ARCHIVERS	2F	25 Frs
DP129	MYDOS 4.5	3F	30 Frs	DP130	DISK CAT 130 XE	1F	25 Frs
DP131	R-DRAW 130 XE	2F	25 Frs	DP132	HI-RES-DUMP 130 XE	2F	25 Frs
DP133	DISK BASE	1F	20 Frs	DP134	SOFTKEYS	1F	20 Frs
DP135	ACTION! UTILITIES 1&2	2F	25 Frs	DP136	NEWSLETTER READER	1F	20 Frs
DP137	AT BASIC POWER PACK	1F	20 Frs	DP138	UTILITAIRES P6	3F	35 Frs
DP139	ARK COMMUNICATION	2F	25 Frs	DP140	TEXTPRO	4F	35 Frs
DP141	PIXEL ARTIST DE LUXE	1F	20 Frs	DP142	FONTMASTER	1F	20 Frs
DP143	THE DIGITAL EDITOR	2F	20 Frs	DP144	WEAK	1F	20 Frs
DP145	ASSEMBLY LANGUAGE TUT.	1F	20 Frs	DP146	DOT MAGIC	3F	30 Frs
DP147	BOB TERM 1.1	2F	20 Frs	DP148	SPARTA UTILS	8F	85 Frs
DP149	DAISY DOT II Fonts	2F	20 Frs	DP150	DRAPER PASCAL	2F	25 Frs
DP151	SPECIAL DATABASE	2F	25 Frs	DP152	MAGIC SPELL	2F	20 Frs
DP153	DISKMASTER	2F	20 Frs	DP154	SUPERDOS 5 / AEB	2F	30 Frs
DP155	PAINT 256	2F	25 Frs	DP156	RAMbrandt Gallery	2F	25 Frs
DP157	ATARI CAD/CAM	2F	30 Frs	DP158	1020 DIGITALIZER	1F	20 Frs
DP159	HOBBY TRONIC Démo'89	2F	25 Frs	DP160	POLISH & TOP #3 ..	2F	35 Frs
DP161	ELVIS Forever	2F	25 Frs	DP162	ATOMIT/MARIO's	2F	30 Frs
DP163	TETRIS 3D/MHDN	2F	30 Frs	DP164	BOULDER DASH/SKI	2F	30 Frs
DP165	The BIG Démo	2F	30 Frs	DP166	BURGER CHEF/BUBBLE	2F	30 Frs
DP167	FUTURA 4 & 5	4F	40 Frs	DP168	DOS XE + MANUEL	5F	45 Frs
DP169	THE NEW DIGITAL EDITOR	2F	30 Frs	DP170	SCREENS	2F	25 Frs
DP171	DISK PICTURE BREAKER	1F	20 Frs	DP172	ASSEMBLY SOURCE IN.	1F	20 Frs
DP173	EASYSTOR	1F	20 Frs	DP174	HACKIN'ABOUT	1F	20 Frs
DP175	PRINT SHOP UTILITIES	1F	20 Frs	DP176	TOUCH EDIT	2F	25 Frs
DP177	CC8 + BIBO DOS	2F	30 Frs	DP178	CC65 (langage C)	6F	50 Frs

OFFRE SPECIALE VALABLE JUSQU'AU 31 MARS 1994

Ref.	*Désignation*	*Prix:*	*Adhérent*	*Non-Adh*
SP004:	Pack Utilitaire (4 Faces) ..... Panachez 4 faces parmis les titres suivants: DP130, DP133, DP134, DP136, DP137, DP142, DP144, DP145		45 Frs	70 Frs
SP005:	R-DRAW + HI-RES-DUMP (4 Faces)		35 Frs	50 Frs

Note: HI-RES-DUMP!, DAISY-DOT II, Graphic Construction Set (DC58), WEAK The New Digital Editor & DOT MAGIC fonctionnent sur imprimante EPSON ou compatible.

Les titres de la série DC01 à DC56 sont des disques simple face et sont vendu 20 Frs pièce ou 30 Frs par deux (de votre choix). Pour ceux de la série DP101 à 153 le nombre de face et le prix sont donnés dans le tableau ci-dessus. Tarifs non-adhérents, rajouter 10 Frs pour les titres 1 face et 15 Frs pour les titres 2 faces Rajouter 9.50 Frs pour les frais de port et d'emballage quelque soit le nombre de titres commandés.

Club CENACLE B.P. 49 95110 SANNOIS

Mes remerciements à tous ceux qui ont oeuvré à la sortie de ce numéro, et principalement à Daniel Carrodano qui a tapé une grande partie des textes. En ce qui concerne les articles et programmes qui n'ont pas pu être mis dans ce numéro par manque de place (ou parce qu'ils me sont arrivés trop tard), rassurez-vous, ils seront publiés dans le prochain numéro.

Si ce numéro est moins copieux que les précédents, nous y avons maintenu les critères de qualité qui ont toujours été les nôtres dans les précédentes éditions. Et, même si la plupart des programmes datent de 1985-86, il faut bien reconnaître qu'ils sont d'un niveau supérieur à tout ce qu'on trouve aujourd'hui dans **PAGE 6**. C'est un fait que si le nombre d'utilisateurs de 8 bits va en décroissant, le nombre de bons programmeurs se restreint aussi. Mais peut-on s'étonner si les meilleurs programmeurs se sont reconvertis sur des machines plus puissantes et plus répandues ?

Ceci dit, le phénomène n'est pas isolé, nombre d'entre vous ont peut-être remarqué que les **Amiga** et **ST** sont aussi en perte de vitesse. Si ces machines ont eu leur heure de gloire dans les années 1987 à 90, elles ne bénéficient plus aujourd'hui du même engouement. On ne débattra pas ici des causes de ce phénomène qui sont bien connues: Le piratage extrême des utilisateurs et le manque de professionnalisme et de dynamisme des distributeurs.

Le marché de la micro-informatique a énormément changé depuis ses premiers pas en 1980. Le concept *d'ordinateur familial* est passé à la trappe, et l'époque où l'on pouvait vendre n'importe quoi à n'importe quel prix est révolue. L'ordinateur personnel est aujourd'hui considéré comme un outil de travail, et, à ce titre, il doit pouvoir s'adapter à tous les besoins des utilisateurs.

Ainsi, l'Amiga, qui a été relégué au rang de console de jeux du fait du manque de programmes sérieux se trouve désormais concurrencé par les consoles de jeux. Quant au **ST**, qui proposait pourtant une gamme de logiciels très diversifiée, il n'a pas su évoluer assez vite pour résister à la concurrence. Et le Falcon, qui était le dernier atout d'Atari, n'est pas parvenu à s'imposer.

Seuls les constructeurs compatibles PC ont su tirer leur épingle du jeu. Et la concurrence qui règne sur ce standard a permis à ces machines d'évoluer à une vitesse phénoménale ces dernières années. Le fait qu'ils s'adressent à une clientèle professionnelle plus exigeante y est certainement pour beaucoup.

Rémi Gallopin

Rédacteur en chef: Rémi Gallopin
Rédacteur: Daniel Carrodano
Ont participé à ce numéro:
Poisson Charles & Perrot Marc.

Sommaire du Cenacle News No 12

Fanzine réalisé sur Word 2.0 pour Windows 3.1 et imprimé sur Laser Olivetti PG 308 PS

PREMIERS PAS

de Mark HUTCHINSON
Traduit de l'anglais par Daniel CARRODANO
(Publié dans New Atari User)

Après avoir sauvegardé et récupéré des écrans, voyons comment en dessiner correctement.

ON PROGRAMME:

La première chose à faire est de se mettre en mode graphique, et un des meilleurs pour ce genre de programme est le GRAPHICS 7 qui a une haute résolution et 4 couleurs. La première ligne de notre programme sera par conséquent:

```
100 GRAPHICS 7
```

La commande COLOR nous autorise à choisir une des quatre couleurs et SETCOLOR nous permet de modifier cette couleur. Pensez que COLOR est un des quatre stylos à encre, et SETCOLOR ne fait que changer l'encre d'un stylo. COLOR 0 est la couleur du fond, et quand un point est PLOTé dans cette couleur il ne peut pas être vu. De plus, utilisé par dessus une autre couleur, cela effacera la couleur originale. Notre prochaine ligne va initialiser une des couleurs.

```
110 COLOR 1
```

Maintenant nous devons imaginer un moyen d'utiliser la manette de jeu pour dessiner quelques jolis motifs. La manette a huit positions possibles, ainsi que démontré en Figure 1.

Nous pouvons lire la position de la manette en utilisant soit la commande STICK, soit en regardant l'emplacement mémoire avec PEEK. J'utiliserai PEEK. Si la manette est poussée vers le haut, la mémoire stockera 14, alors nous dirons à l'ordinateur que s'il trouve un 14, il doit garder X comme valeur présente (nous n'avons pas bougé latéralement), et soustraire 1 de Y pour faire monter le curseur. Ces coordonnées d'écran sont montrées en Figure 2.

Si vous regardez l'écran, le haut à gauche est la position du point zéro. En se déplaçant de gauche à droite on trouve les coordonnées de X, et de haut en bas les coordonnées de Y. Pour aller vers le haut nous soustrayons 1 à Y, pour aller vers le bas nous ajoutons 1 à Y. Pour aller à droite, nous ajoutons 1 à X, et pour aller à gauche, nous ôtons 1 de X. C'est simple.

Notre nouvelle ligne sera:

```
120 IF PEEK(632)=14 THEN Y=Y-1
```

Mais nous aurions à écrire huit lignes semblables, une pour chaque position de la manette. Il y a un meilleur moyen. Comme vous pouvez l'imaginer il n'y a jamais qu'une seule valeur à tout moment donné, et il est possible, en utilisant l'algèbre booléenne, de combiner toutes les conditions variables dans une seule ligne. Pas de panique, pas la peine de sortir de l'université pour ça; c'est vraiment très simple.

UN PEU D'ALGEBRE BOOLEENNE

Si une formulation telle que (P=14) est correcte (VRAI), alors on obtient une valeur de 1. Si ce n'est pas correct (PAS VRAI) alors ce sera 0. Donc, une autre façon d'écrire la ligne serait:

```
120 Y=Y-(PEEK(632)=14)
```

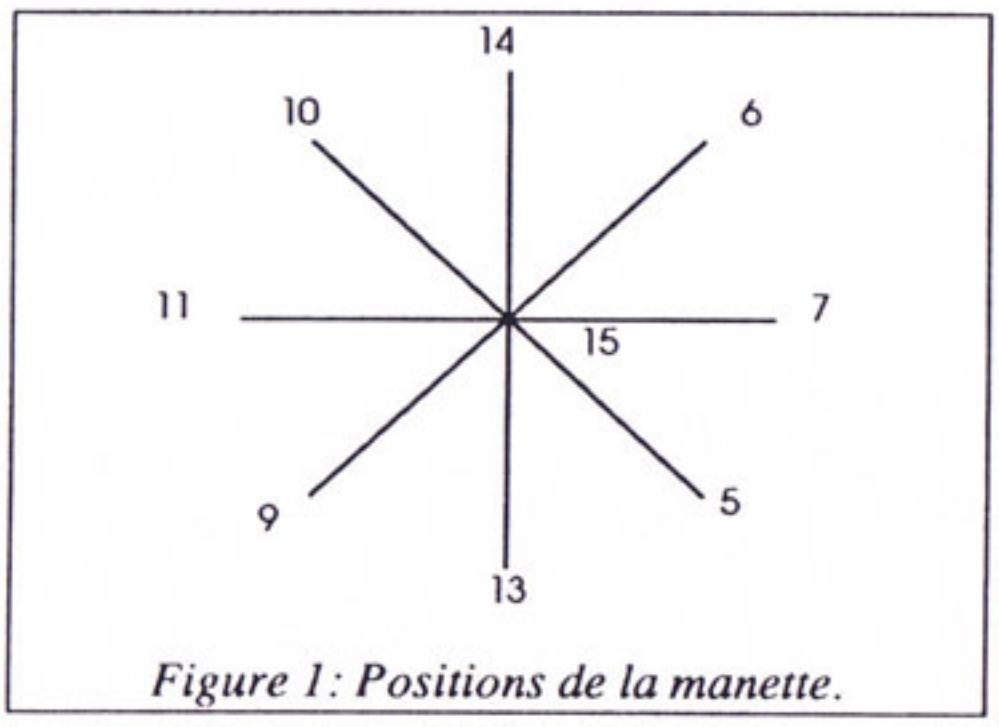


Figure 1: Positions de la manette.

Quand le PEEK(632) est égal à 14, il est ôté de Y comme précédemment, mais s'il n'est pas égal à 14 rien n'est retiré. Cette ligne parait compliquée alors je vais la changer pour la rendre plus lisible et je vais rajouter le reste des valeurs de la manette de la même manière.

```
120 P=PEEK(632)
130 Y=Y+(P=5)+(P=9)+(P=13)
140 Y=Y-(P=6)-(P=10)-(P=14)
150 X=X+(P=5)+(P=6)+(P=7)
160 X=X-(P=9)-(P=10)-(P=11)
```

J'ai divisé la routine en cinq lignes pour plus de clarté, mais ça peut tout tenir sur une seule ligne. Examinons un peu cette routine. Depuis le diagramme 2 nous pouvons voir que X sera implémenté quand la manette retournera la valeur 5, 6 ou 7. Si c'est 5, la première condition entre parenthèses sera vraie, par conséquent à 1, et le reste sera à zéro. Ainsi, 1 sera ajouté à X. Si la valeur est 13, aucune modification ne sera apportée à X. Aucune des conditions ne sera vraie et toutes les valeurs seront à 0. Pour avoir quelque chose à l'écran nous utiliserons la commande: PLOT X,Y.

Reprenons notre exemple et rajoutons ceci:

```
170 PLOT X,Y
180 GOTO 120
```

Pour l'instant nous n'avons utilisé qu'une seule couleur. Nous pourrions utiliser le clavier pour choisir une autre couleur, mais regardons plutôt la case 644, qui gère le bouton de la manette, et modifions la couleur en utilisant cela. La raison est que, quand le bouton est pressé, une valeur de 0 est retournée, et j'ai cherché à vous montrer le contraire des lignes 130 à 160. Notez que nous devons alors déplacer GOTO & PLOT.

```
110 COLOR C
170 C=C+NOT PEEK(644)
180 IF C>3 THEN C=0
190 PLOT X,Y
200 GOTO 110
```

En ligne 170, C sera implémenté d'une valeur de 4. Comme la commande COLOR ne peut reconnaître qu'une valeur de 0 à 3, la ligne 180 remettra automatiquement COLOR à 0, la couleur du fond. Il est facile de tester et annuler X et Y de la même façon qu'en ligne 180. Si X ou Y passent hors de la bordure d'écran, il se produira une erreur 141, et nous devons en tenir compte. Si, par exemple, Y atteint le bas de l'écran, nous devons réinitialiser le curseur pour qu'il apparraisse en haut. Ceci s'appelle "wraparound". 159 et 79 sont les coordonnées de l'écran en mode 7 avec fenêtre; nous avons donc maintenant:

```
190 IF X>159 THEN X=0
200 IF X<0 THEN X=159
210 IF Y>79 THEN Y=0
220 IF Y<0 THEN Y=79
230 PLOT X,Y
240 GOTO 110
```

C'est le programme de base et je vous laisse le modifier à votre façon. Par exemple une certaine action sur le clavier peut envoyer le programme à la routine SAVE ou LOAD que nous avons déjà étudié. Unez dernière chose, rajoutez cette ligne avant PLOT.

PRINT "X= ";X;" Y= ";Y

Cela vous permet de mieux apprécier la position du curseur. Voilà pourquoi j'ai gardé la fenêtre de texte, mais vous ne souhaitez peut-être pas garder la fenêtre avant de sauvegarder, alors essayez cette ligne avant:

```
GRAPHICS 7+16+32 (ou GR. 55)
```

Cette ligne garde le même mode graphique (7), le 16 supprime la fenêtre de texte et le 32 gardera ce que vous aviez dessiné originalement. Je n'ai pas vu cette utilisation depuis des années, mais c'est un bon exemple pour débuter.

Les routines SAVE/LOAD que je vous avais donné pourront sauvegarder tous les modes graphiques sur disquette ou cassette ou être utilisées pour transférer de la RAM ou des jeux de caractères. Si vous regardez le listing 2 précédent, vous noterez les offsets 4 et 5. Ils gèrent l'emplacement mémoire que vous désirez lire ou écrire. Le caractère sera sauvegardé ou chargé un octet à la fois, donc la routine IOCB sera utilisée pour un accès plus rapide.

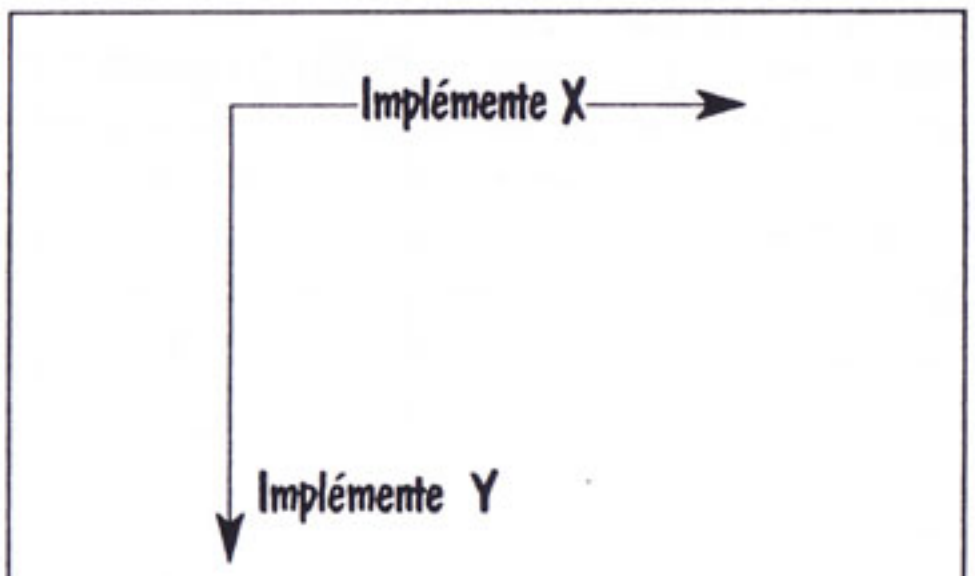


Figure 2: Coordonnées de l'écran.

LA DISPLAY LIST, COMMENT ÇA MARCHE ?

Par Rémi Gallopin

ANTIC, qui gère l'affichage l'affiche du XL/XE, utilise une sorte de langage machine primaire (en comparaison du 6502C), c'est la Display List (DL) qui contient ce "programme". Cette dernière se trouve juste en dessous de la mémoire écran. Elle débute à l'adresse donnée par:

DL=PEEK(560)+256*PEEK(561)

Pour afficher une DL d'un écran Graphics 0, essaye ce petit programme.

```
10 GR. 0 : DL=PEEK(560)+256*PEEK(561)
20 FOR X=DL TO DL+31 : PRINT X ,
PEEK(X) : NEXT X
```

Ce qui donne la liste suivante :

112 112 112 66 64 156 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
65 32 156

Pour mémoire, un écran Atari est constitué de "Scan Lines" en anglais dans le texte. Une Scan line correspond à une ligne de mode GR 8. La marge du haut de l'écran est constituée d'environ 32 scan lines sur lesquelles on ne peut rien afficher. Vient ensuite la zone affichable qui est de 192 scan lines, quelque soit le mode graphique choisi. Puis il reste un certain nombre de scan lines qui font la marge du bas.

Bon mais revenons-en à notre DL. Les trois premiers 112 correspondent à la marge du haut (chaque 112 indique à Antic de générer 8 scan lines vierges qui ne seront pas utilisables et auront la couleur de la bordure - (COLOR4). Dans le tableau 2, il y a les autres instructions qui correspondent à des lignes de même type mais d'un nombre de scan lines différentes.

Puis le 66 suivant doit se lire:

64 + n (ici n=2)

64 indique à Antic d'aller lire la zone de mémoire écran dont l'adresse sera déterminée par les deux octets qui suivent : 64 et 156
Ce qui donne l'adresse mémoire suivante: 64 + 256 * 156 - les adresses mémoire étant toujours données sous la forme octet de poids faible / octet de poids fort - note qu'on retrouve aussi l'adresse écran en faisant Peek(88) + 256 * Peek(89) puisque ce pointeur indique au Basic le début de la mémoire écran pour les instructions PRINT et LOCATE entre autres.

n indique quel est le mode graphique de la première ligne de l'écran, mais attention, la numérotation des modes graphiques Antic est différente de celle du Basic. Ainsi 2 correspond au mode GR. 0 (pour les autres modes, voir le tableau 1).

Puis, on trouve 23 fois le chiffre "2". Ce sont les 23 lignes suivantes de l'écran mode 0.

Enfin, tu as 65 qui dit à Antic de sauter à l'adresse donnée dans les 2 octets suivants (32 + 256 *156), après un vertical Blank - toujours en anglais dans le texte.

Note encore que cette adresse correspond à DL - le début de la Display List.

Alors un Vertical Blank, kézaco ???

C'est simple, l'affichage est effectué par un spot qui balaie l'écran tous les 50° de seconde. Donc quand le spot est arrivé au coin inférieur droit, il doit remonter au coin supérieur gauche. C'est le temps nécessaire à ce déplacement qui est appelé Vertical Blank ! Ainsi l'instruction 65 indique à Antic de se synchroniser avec le spot avant d'envoyer la trame suivante à l'écran.

Voici donc comment ça marche en théorie une Display List.

Le principe mis en oeuvre par Atari pour gérer ses écrans est à la fois simple et puissant, puisqu'il ouvre la porte à de nombreuses bidouilles.

A partir de là, on voit qu'il est très facile de créer des écrans mixtes sur l'Atari, il suffit de modifier la DL. Et on pourra ainsi obtenir un écran contenant des lignes en mode 0, en mode 1 et en mode 2 en même temps par exemple. Ou même mixer des modes graphiques avec des modes textes. La procédure est simple, tu fais une instruction GRAPHICS avec le mode occupant le plus de mémoire parmi ceux que tu as retenu pour ton écran.

Puis avec des Pokes, tu ajustes le DL, en faisant gaffe à pas te planter! Car il ne faut pas perdre de vue que l'Atari met son écran tout en haut de la mémoire (en dessous l'adresse donnée par PEEK(106) * 256). Et la DL juste en dessous de la mémoire écran.
Donc il faut être sur que la mémoire écran sera assez grande pour ce que tu as mis dans la DL.
Une autre règle, le nombre de scan lines de ton écran est limité à un maximum de 232 environ, ainsi en diminuant la taille de la marge du haut, tu peux augmenter la taille de la zone d'affichage au-delà de 192 scan lines - je crois me rappeler que dans le CN 1, Laurent Decomble avait donné une petite routine qui permettait d'obtenir 29 lignes utilisables en mode GR. 0.

ANTIC Mode	BASIC Mode	Scan lines / Mode lines	Bytes/ lines	Screen Memory	Total Memory
2	0	8	40	960	992
3	-	10	40	760	-
4	12	8	40	960	1152
5	13	16	40	480	660
6	1	8	20	480	672
7	2	16	20	240	420
8	3	8	10	240	432
9	4	4	10	480	696
10 (A)	5	4	20	960	1176
11 (B)	6	2	20	1920	2184
12 (C)	14	1	20	3840	4296
13 (D)	7	2	40	3840	4200
14 (E)	15	1	40	7680	8138
15 (F)	8 (9,10,11)	1	40	7680	8138

Tableau 1: Modes graphiques de l'Atari

ANTIC command	Nbre Scan Lines
112	8
96	7
80	6
64	5
48	4
32	3
16	2
0	1
ANTIC command	**Fonction**
1	Jump to adress
65	Jump at VBI

Tableau 2 : Instructions Antic

Etiquette	adresse décimale	Etiquette	Registre Antic
PCOLR0	704	COLPM0	53266
PCOLR1	705	COLPM1	53267
PCOLR2	706	COLPM2	53268
PCOLR3	707	COLPM3	53269
COLOR0	708	COLPF0	53270
COLOR1	709	COLPF1	53271
COLOR2	710	COLPF2	53272
COLOR3	711	COLPF3	53273
COLOR4	712	COLBK	53274

Tableau 3: Registres de couleurs

COMMENT AVOIR PLUS DE COULEURS A L'ECRAN ?

Un petit rappel: On choisit les couleurs d'affichage par SETCOLOR ou POKE aux adresses 708 à 712 (712 étant la bordure - et aussi le fond en mode autres que 0). Et au moment de l'affichage, le contenu de ces adresses est recopié dans les registres hardware de Antic. Dans le **tableau 3**, tu retrouves ces registres sous les noms COLPF0 à COLPF3 et COLBK avec leur équivalent en mémoire basse (708 à 712) - voir aussi PROGRAMMER'S CARD en fin du DE RE .ATARI

Donc l'astuce est de changer le contenu de ces registres entre le moment où ANTIC a commencé à construire son écran et avant la fin de l'affichage. Comme ça Antic affichera la première partie de l'écran avec les couleurs mises aux adresses 708 à 712, puis la deuxième partie avec les nouvelles valeurs qu'on aura mises dans les registres Antic. Avec ce principe, tu doubles le nombre de couleurs, et Antic n'y voit que du feu! Il peut aussi être possible de faire cette opération plusieurs fois dans un même écran, mais c'est un peu plus complexe à mettre en oeuvre. Il va s'en dire que cette procédure ne pourra se faire qu'en assembleur, car elle doit être synchronisée avec l'affichage d'un écran qui se fait en moins de 1/50 de seconde! Maintenant, cette méthode à ses limites, puisque à un endroit donné de l'écran, tu n'auras jamais plus que le nombre légal de couleurs dans le mode choisi (de 2 en GR. 0 à 5 en GR. 1 & 2).

La mise en place d'une DLI se décompose en trois étapes: 1 - écrire le programme en assembleur qui exécutera cette DLI, 2 - Placer cette routine en mémoire, et 3 - enfin autoriser la reconnaissance de cette DLI.

Reprenons ces étapes une par une:

Le programme assembleur doit commencer par sauvegarder dans la pile tous les registres du 6502 qui vont être utilisés. Ce programme doit être le plus court possible, car il ne dispose pas d'un nombre de cycles énorme. Puis il se terminera en restituant les registres à partir de la pile.

Voyons l'exemple d'un programme très simple qui va modifier la couleur de fond d'écran de bleu en rose à partir du milieu d'écran. Pour garder une bonne lisibilité, la couleur des caractères sera aussi modifiée. Etant dans le mode graphique zéro, nous travaillerons donc sur les registres COLOR1 et COLOR2 ainsi que leurs équivalents dans ANTIC COLPF1 et COLPF2 .

Le listing assembleur sera donc:

```
10 COLPF1=$D017
20 COLPF2=$D018
30 WSYNC=$D40A
90  *=$600
100  PHA
110  TXA
120  PHA
130  LDA $#50
140  LDX $#58
150  STA  WSYNC
160  STA  COLPF1
170  STX  COLPF2
180  PLA
190  TAX
200  PLA
210  RTI
```

Explicitons ce programme:

PHA pousse le contenu du registre Accumulateur dans le pile.

TXA transfert le contenu du registre X dans le registre A. On utilise TXA suivit d'un PHA pour stocker le contenu de X dans la pile, car il n'existe pas d'instruction transférant directement le contenu de X dans la pile.

On charge ensuite les registres X et A avec les valeurs correspondant aux couleurs que l'on veut obtenir. Et ces valeurs sont ensuite stockées par STA et STX dans les registres ANTIC COLPF1 et COLPF2.

L'instruction STA WSYNC est obligatoire, elle permet une synchronisation avec l'affichage - en fait, il suffit de stocker n'importe quelle valeur dans WSYNC.

Puis, on restitue les contenus originels de A et X par PLA qui pousse la pile dans A et TAX qui pousse A dans X (idem qu'en début).

La routine doit impérativement se terminer par une instruction RTI.

Une fois ce programme assemblé, on transformera le code objet en lignes de DATA Basic (voir les routines fournies à cet effet dans le Cenacle News 10). Ce qui nous permet d'obtenir le programme Basic suivant:

```
10 DLI=PEEK(560)+256*PEEK(561)
20 POKE DLI+15,130
30 FOR I=0 TO 19
40 READ A:POKE 1536+I,A:NEXT I
50 DATA 72,138, 72,169, 80,162, 88
60 DATA 141, 10,212,141, 23,208
70 DATA 142, 24,208,104,170,104, 64
80 POKE 512,0:POKE 513,6
90 POKE 54286,192
```

La ligne 20 indique la ligne écran à partir de laquelle la modification de couleur doit se programmer, par un positionnement du bit 7 à 1 (voir explicatif en 1ère partie d'article). les lignes 30 à 70 chargent la routine en Page 6, la ligne 80 indique l'adresse de cette routine à ANTIC. Et pour terminer, on positionne à 1 le bit 7 de l'adresse $D40E (NMIEN) pour autoriser la DLI (ligne 90). L'interruption sera donc validée à partir de cet instant et seul une instruction GR. x pourra l'arrêter.

On pourra se demander pourquoi la couleur d'écran reste bleue en haut. C'est parce que l'O.S., avant chaque début d'écran, replace les valeurs des registres COLOR0 à 4 dans les registres ANTIC COLPF0 à 3 et COLBK. Et notre routine les remodifie en cours d'affichage tout les 50ème de seconde.

Donc, dans la pratique, une fois que tu as décidé du type d'écran qu'il te faut, tu repères à partir de quelle ligne d'écran il te faut le changement des registres de couleur. Et sur la ligne précédente, tu ajoutes 128 au code Antic du mode graphique. Prenons l'exemple de la DL du mode GR. 0 que je donne en début d'article, pour obtenir un changement de couleur à la 10ème ligne écran, il te faudra remplacer le 8ème "2" par 130. Eh oui, ce "2" là correspond à la neuvième ligne d'écran, car rappelle-toi, la première ligne écran est générée par le 66 et non par le 1er "2".
Une fois cela fait, il faut indiquer à Antic l'adresse de ta routine assembleur par: POKE 512, ADRLOW : POKE 513, ADRHIGH
Puis tu fais POKE 54286, 192 pour dire à Antic qu'il faut exécuter le DLI. Voilà...

Maintenant, si on voulait changer de couleurs plusieurs fois par écran, ça reste possible. On mettrait à 1 le bit 7 correspondant à plusieurs lignes d'écran, et la routine serait déclenchée après chacune de ces lignes. Seulement, comme on ne peut avoir qu'une seule routine machine, elle devrait gérer tout les changements. Il faudrait donc intégrer un compteur pour savoir à quel endroit de l'écran on se trouve... et envoyer les bon codes de couleurs. Bref, ça complique un peu la sauce.

Pour vous simplifier la vie, reportez-vous au programme **DLI Maker** décris plus loin dans ce numéro. Il permet de générer automatiquement des DLI adaptées à vos besoins.

Suite de l'article DLI-Maker

16. Use Graphic mode
17. 0
18. Place on line 8 and 9

voir Programme DLIX2.BAS

C. Mirroir, mirroir:

1. Change a memory location
2. Some other address
   (Quelques autres adresses)
3. $D401
4. To a Constant (en une constante)
5. 6
6. Sauvegarde DLI et quitte
7. Use Display List byte number
   (Utilise octet display list)
8. Place on line 16
   (Place sur octet 16)

voir Programme DLIX3.BAS

DLI MAKER

Par Greg ANDERSON

Article paru dans ANALOG COMPUTING de janvier 1986

traduit de l'américain par Daniel CARRODANO

mis en page par Rémy GALLOPIN.

Des DLI personnalisées pour vos programmes BASIC.

N'avez-vous jamais désiré insérer une interruption de Display List dans un de vos programmes BASIC, en y renonçant à cause de votre méconnaissance du langage machine? Vos jours de désespoir sont terminés avec DLI Maker! Vous n'avez pas besoin de vous appeler Tom Hudson pour insérer de multiples DLI dans un programme BASIC, **DLI Maker** le fera entièrement pour vous! **DLI Maker** est un programme BASIC qui va créer un jeu de DLI personnalisé très facile à incorporer à vos programmes. Il vous suffit de progresser à travers les menus pour vous fabriquer une DLI particulière en quelques minutes, sans le recours à un assembleur. Même les programmeurs expérimentés en langage machine trouveront DLI Maker plus simple que d'écrire et déboguer leurs propres DLI avec un assembleur. En fait, DLI Maker LISTera automatiquement un fichier sur disque ou cassette, complet avec toutes les initialisations requises, de ce fait il n'y a nul besoin de convertir manuellement un fichier code objet en code BASIC utilisable.

C'est quoi une **DLI**, pouvez-vous demander? (*NDLR: voir article sur ce sujet dans ce même Cenacle News*) Pour simplifier, une DLI vous permet de modifier la formation de l'image à l'écran. Par exemple, vous pouvez désirer que les dix premières lignes du haut d'un écran Graphique 0 soient avec un fond bleu et que les quatorze lignes plus basses soient noires. Ou vous pouvez souhaiter utiliser un jeu de caractères particuliers dans la partie Graphique 1 de l'écran, tout en ayant le jeu de caractères standard dans la fenêtre texte. Naturellement, il n'y a pas d'obligation à vous limiter à un seul changement par écran. Il est possible de faire plusieurs modifications dans chaque ligne de l'écran. Jusqu'à présent, si vous vouliez faire quelque chose dans ce genre, vous n'aviez pas d'autre moyen que d'écrire votre propre DLI en langage machine. Par contre, aucune connaissance du langage machine n'est nécessaire pour utiliser **DLI Maker**: tout le "sale boulot" est fait pour vous. La connaissance des PEEKS et POKES est utile pour la création d'une DLI, mais pas nécessaire pour réaliser les plus simples. **DLI Maker** est très simple d'emploi. Il vous suffit de circuler à travers plusieurs menus. Quand vous en avez terminé, le programme listera votre DLI sur disque ou cassette dans une forme qui pourra ensuite être récupéré avec ENTER dans vos programmes BASIC. Les numéros de ligne utilisés vont de 10000 à 11200. Tout ce que votre programme a besoin de faire c'est un GOSUB à la ligne 11000 lors de son initialisation, et GOSUB 10000 après les commandes graphiques, pour mettre en action vos DLIs. Attention: les DLIs sont stockées dans une variable chaîne, ce qui fait qu'elles doivent être supprimées (avec une commande graphique ou une remise à zéro du système par RESET) avant que le contrôle ne soit rendu au BASIC. Les chaînes perdurent aussi longtemps que votre programme tourne, mais le BASIC les manipule librement quand vous entrez d'autres lignes de BASIC ou quand vous exécutez une commande en mode direct. Si les DLIs sont activées pendant que les chaînes sont manipulées, l'ordinateur plantera et vous n'aurez pas d'autre choix que de l'éteindre et le rallumer. Pour éviter cela, quittez votre programme en utilisant RESET plutôt que la touche BREAK.

Fabrication d'une DLI

Quand vous faites tourner **DLI Maker** pour la première fois, et au début de chaque pas de chaque DLI, le menu principal vous sera présenté. C'est le premier des nombreux menus que vous utiliserez pour indiquer à **DLI Maker** exactement ce que vous désirez que votre DLI accomplisse.

Changement d'une case mémoire

Alors qu'il y a au total cinq sélections différentes dans le menu principal, la plupart des DLIs n'utiliseront que le premier, "Change a memory location". Bien qu'il soit plus simple de vous demander tout bêtement la case mémoire que vous désirez changer, **DLI Maker** tient plusieurs emplacements mémorisés à votre disposition, tels que couleurs d'écran, et pointeur de jeu de caractères. Si DLI ne connaît pas la case mémoire que vous désirez modifier, vous pouvez la taper, tant en décimal qu'en hexadécimal, la dernière valeur précédée d'un signe $, par exemple $400. Vous êtes sans doute déjà familiarisés avec les adresses mémoire de 708 à 712, qui sont les registres de couleur (modifiables respectivement par SETCOLOR 0 à 4). Ces registres correspondent directement aux registres hardware de **ANTIC** ($D016 à $D01A), dont ils sont les "ombres", ou les duplicatas en RAM.

Alors, qu'elle différence y-a-t'il entre les registres hardware de **ANTIC** et les registres "ombres" en RAM, si les deux contrôlent la même chose? *Les "ombres" de registres peuvent être lus et écrits, tandis que les registres hardware sont soit en lecture soit en écriture seulement*. C'est un concept très important. Les registres hardware contrôlent normalement les opérations données, mais, du fait qu'ils ne peuvent pas être lus une fois qu'ils ont été écrits, le SE (système d'exploitation) de l'Atari maintient les "ombres" de registres à la disposition du programmeur. Ces emplacements sont copiés dans leurs emplacements hardware correspondant cinquante fois à la seconde (en Europe), au début du cycle de dessin de l'écran. Dans une DLI, vous **devez êtres sûrs** de changer les registres hardware adéquats, parce que, si vous en changez les ombres, vous modifierez l'apparence de l'écran complet, pas seulement la partie dessous la DLI. Le plus grand avantage des "ombres" de registres, est que vous pouvez changer la couleur de l'écran au milieu d'un écran, et l'ombre du registre maintiendra constante la couleur du haut. Hélas, les registres hardware n'ont pas tous une ombre; par exemple les registres de position horizontale des lutins (players/missiles), qui vont de $D000 à $D007. Si vous changez ceux-ci à un endroit de l'écran, vous devez aussi avoir une DLI dans la première ligne, de façon que vos lutins ne soient incorrectement positionnés au-dessus de votre première DLI. Assez souvent vous voudrez changer la case mémoire sélectionnée en une constante (le même nombre à chaque fois). De nouveau, le nombre peut être entré en décimal où en héxa, ou, si une couleur d'écran est en train de changer, **DLI Maker** vous affichera la liste des seize couleurs Atari, suivies de leurs huit luminances. Occasionnellement, il est plus simple de changer une case mémoire par la valeur même stockée à une autre adresse, plutôt que d'utiliser une constante. Vous pouvez, par exemple, vouloir changer la couleur de la moitié haute de l'écran dynamiquement à l'intérieur de votre programme BASIC. Lors de la sélection de copie d'une adresse sur une autre, vous n'avez pas d'autre choix que de taper la valeur de case mémoire en question, du fait que tout ce que **DLI Maker** sait au sujet des valeurs concerne les registres hardware Si vous accomplissez plus d'un pas sur la même DLI, il vous sera donné l'option supplémentaire permettant de changer une case mémoire dans la même valeur au pas suivant. Cela est pratique si vous voulez mettre le fond et la bordure d'un écran en Graphique 0 dans la même couleur. Notez que ceci ne fonctionne que pour un changement ou un ajout vers une case mémoire. Augmentation, diminution et attente de syncro horizontale n'affectent en aucune façon l'Accumulateur du 6502, lequel gère les dernières valeurs stockées. Par conséquent, le changement de la couleur du fond en vert, suivi d'une augmentation, suivie d'un changement de la bordure dans la couleur du dernier pas changera les deux couleurs, fond et bordure, en vert. La bordure ne fluctuera pas, comme on pourrait s'y attendre de l'augmentation précédente.

Ajout vers une case mémoire

Dans de rares circonstances, vous pouvez désirer faire quelques additions ou soustractions à l'intérieur d'une DLI.. La soustraction est réalisée par l'ajout d'un négatif: **DLI Maker** convertira automatiquement les nombres négatifs en leurs deux compléments équivalents. L'ajout de cases mémoires est plus complexe que les autres fonctions de DLI. C'est similaire au

changement d'une case, mais vous devez aussi spécifier deux valeurs à ajouter, dont une des deux sera toujours une case mémoire. La première chose que l'ajout vous demandera est l'adresse dans laquelle sauvegarder le résultat. Puis, la constante (ou seconde adresse d'un ajout) est requise, suivie par la case mémoire à laquelle elle est ajoutée. Pressez RETURN au dernier paramètre pour utiliser la même case comme résultat. Deux autres questions sont aussi posées: "**Binary or BCD?**" et "**Clear carry?**" Elles sont incluses pour les programmeurs en langage machine qui peuvent ne pas vouloir utiliser les conditions par défaut. Mais vous si, alors pressez RETURN à ces questions. Faites particulièrement attention de ne rien ajouter depuis un registre matériel. Souvenez-vous, la valeur lue sur un registre matériel est différente de la dernière valeur stockée.

Augmentation et diminution

Augmentation ou diminution (ajout ou retranchement de 1 à une case mémoire) sont aussi possibles. De nouveau, rappelez-vous que les registres hardware ne peuvent pas être incrémentés directement. Vous devez à la place réserver un emplacement quelque part, l'augmenter et changer le registre hardware dans sa valeur au prochain pas. Lors de l'utilisation d'espace supplémentaire comme variables dans une DLI, vous devez être sélectif au sujet des adresses à utiliser. Il est important que vous choisissiez une adresse dont vous savez qu'elle ne sera pas modifiée par toute autre routine. Vous ne pourriez pas utiliser une des variables de la page 0, (mémoire comprise entre 0 et 255), car trop peu d'elles sont inemployées. De même, la page 6 (mémoire allant de 1536 à 1791) devrait être évitée. Bien que cette aire soit techniquement libre, trop de programmes BASIC la remplissent de routines en langage machine non relogeables. Donc gardez autant que possible la page 6 libre pour éviter les conflits. Le meilleur endroit pour les variables DLI se trouve dans l'aire mémoire de la pile du 6502 (256-511). La pile du 6502 débute à la case 511 et se développe en descendant. Elle ne se trouve que très rarement plus qu'à moitié pleine, donc vous avez plus de 100 octets disponibles, débutant à l'adresse 257. Et alors pour 256? Les DLI construites par DLI Maker utilisent la case 256 comme un compteur, de ce fait vous ne devez pas y toucher, sinon vos DLI ne tourneront pas correctement.

Attente de syncro horizontale

L'attente de syncro horizontale compile simplement un **STA WSYNC ($D40A)** dans votre DLI. Ceci a pour effet d'arrêter le microprocesseur 6502 jusqu'à ce que le faisceau des électrons qui dessine l'image de l'écran atteigne l'extrême droite. Cette commande est utilisée pour s'assurer que les modifications de couleur ou autres se produisent proprement à l'écran au moment où elles ne sont pas visibles. Il en résulte que le prochain changement de votre DLI se produira une ligne de balayage plus bas (ce qui est de la même largeur qu'une ligne de mode Graphique 8). **DLI Maker** insère automatiquement un STA WSYNC au début de chaque DLI, ce qui fait que vous aurez rarement à utiliser cette commande. Que devais-je faire alors?

Pratiquement à chaque prompt, vous pouvez répondre par X pour vérifier la DLI sur laquelle vous êtes en train de travailler. (Les exceptions sont les questions sur un caractère "*Clear carry*" et "*Binary or BCD*" posées en plus). Le listing marquera une pause après chaque page et attendra que vous pressiez une touche, donc rien ne sera perdu en haut lorsque vous voudrez revoir de longues DLIs.

Terminer une DLI.

Après avoir complété chaque pas de votre DLI, vous devrez dire à **DLI Maker** ce que vous entendez accomplir ensuite. Vos choix sont les suivants: En premier, vous pouvez ajouter un autre pas à cette DLI. Chaque pas de chaque DLI se produira sur la même ligne d'écran (à moins que vous insériez des commandes d'attente). En second, vous pouvez faire une autre DLI. Les DLIs additionnelles se produisent plus loin dans le bas du même écran. **DLI Maker** ne créera pas automatiquement plusieurs DLIs pour des écrans différents. Votre troisième choix est de sauvegarder la DLI que vous venez juste de terminer. **DLI Maker** vous demandera alors dans quel mode ou ligne(s) graphi-

que(s) insérer votre (vos) DLI(s) et la (les) listera sur disque ou cassette. Les deux dernières options sont des corrections d'erreur. Vous pouvez rappeler la DLI sur laquelle vous travailliez précédemment (tous les pas), ou vous pouvez repartir à zéro. Aucune autre commande d'édition n'est reconnue.

Positionnement des DLIs à l'écran

Quand vous aurez créé votre DLI particulière, il restera encore le problème de son positionnement à l'écran. Pour une plus grande flexibilité, **DLI Maker** vous donne la possibilité de choisir la position de chaque DLI soit en numéro de lignes graphiques BASIC, soit en numéro d'octets de display list ANTIC.

Lors de l'entrée de positions de DLI en utilisant les méthodes précédentes, tous les nombres représentent la première ligne où le changement devra se produire. Par exemple, si vous créez une DLI qui change la couleur 1 rouge et la placez sur la Ligne 10 d'un écran Graphique 7, alors tous les pixels de la couleur 1 plottés depuis les Lignes 0 à 9 seront jaunes (à moins d'être SETCOLORisés en quelque chose d'autre), et toute la Ligne 10 et les suivantes seront rouges.

La seconde méthode n'est vraiment utile que si vous utilisez une display list personnalisée. Tous les numéros d'octets entrés sont les octets exacts que la DLI initialisera, donc la DLI se produira dans le mode de ligne suivant. Pour accomplir le même effet que dans l'exemple suivant, la DLI devrait être placée dans l'octet 13 (le premier octet a le numéro 0).

Les DLIs sont sans doute la caractéristique la plus puissante des ordinateurs domestiques Atari (avec la possible exception des player/missile graphiques). Vous trouverez que les DLIs ouvrent tout un nouveau monde à la programmation. Avec elles, vous pouvez doubler ou tripler le nombre de couleurs affichées dans un écran, ou même afficher les 128 couleurs toutes d'un coup! La différence que de nombreuses couleurs peuvent apporter à un écran est fantastique. **DLI Maker** est sûr de trouver une place permanente dans votre logithèque d'utilitaires.

Exemples de DLIs:

Suivez les instructions pour créer une DLI, puis entrez-la depuis le disque avec ENTER. Ajoutez le code listé dessous la DLI. Faites tourner le programme avec RUN, et appréciez.

A. Mélange de jeux de caractères:

1. Change memoire location
   (change une case mémoire)
2. The Character set
   (le jeu de caractère)
3. To a Constant
   (En une constante)
4. $E2
5. Save DLI and QUIT
   (Sauvegarde la DLI et quitte)
6. Use Graphics mode
   (Utilise mode graphique)
7. 18
8. Place on line 5
   (Place en Ligne 5)

voir Programme DLIX1.BAS

B. Ligne pulsante:

1. Incrément memory location
   (Incrémente emplacement mémoire)
2. Adresse $101
3. Add Another step
   (Ajoute un autre pas)
4. Change a memory location
   (Change une case mémoire)
5. A Screen Color
   (Une couleur d'écran)
6. Graphique 0 et luminance 8
7. To a value of another address
   (Vers la valeur d'une autre adresse)
8. $101
9. Do another DLI
   (Faire une autre DLI)
10. Change a memory location
    (Change une case mémoire)
11. A Screen Color
    (Une couleur d'écran)
12. Graphique 0 et luminance 8
13. To a value of another address
    (Vers la valeur d'une autre adresse)
14. 709
15. Sauvegarde DLI et quitte

Suite de l'article page 7

SMARTRAM 2.5

Super RAMdisk pour tous les XL/XE étendus

Par Tim PATRICK, ANTIC Publishing

Traduit par Daniel CARRODANO

Tirez avantage de votre extension mémoire avec **SMARTRAM 2.5**. Ce super RAMdisk tourne sur Atari XL et XE ayant un minimum de 128 Ko de mémoire et le DOS 2.5.

Les nombreuses variantes d'extension mémoire pour XL/XE présentent quelques différences entre elles, mais aussi de fortes analogies. Dans cet article voyons comment sont faites ces extensions et apprenons à communiquer avec elles.

- SmartRAM 2.5 est un puissant disque virtuel qui travaille avec votre extension mémoire, quelle que soit la façon dont elle a été installée. Vous trouverez SmartRAM beaucoup plus pratique et convivial que l'officiel RAMDISK.COM Atari. Pour une meilleure compréhension des différentes extensions de RAM existantes, vous pouvez vous reporter au livre "DR. Brilliant's Incredible Atari Brain Transplants", critiqué dans le numéro de novembre 1988 d'ANTIC.

- Toutefois, SmartRAM n'est pas compatible avec les Atari 800 étendus (NDLR: cartouche AXLON).

- Toutes les extensions XL/XE sont très similaires. Il n'est pas très difficile de les distinguer et c'est un jeu de communiquer avec elles.

- Vous pouvez contrôler chaque extension XL/XE courante avec simplement deux bits de code.

- SmartRAM 2.5 nécessite le DOS 2.5 Atari. Bootez avec une copie de DOS 2.5 et lancez le programme SMARTRAM.BAS. Ce programme écrit un fichier RAMDISK.COM qui vient recouvrir l'original sur la disquette; pensez donc à le déverrouiller si nécessaire. Votre nouveau RAMdisk est prêt à être utilisé. Rebootez en appuyant sur [RESET].

- Note aux utilisateurs: Je n'ai pas fait de test avec le BASIC XE de OSS/ICD, mais celui-ci devrait cependant vous autoriser sans problème le mode d'extension avec un RAMDISK 1050. SmartRAM tourne aussi avec les kits XE-GM1 et XE-GM2 d'Innovative Concepts.

BANQUES D'ADRESSES.

Situé dans le microprocesseur 6502, le registre "compteur" dit à l'ordinateur où aller en mémoire pour lire les instructions ou les données. Ce compteur est dimensionné sur 16 bits. Avec 16 bits on peut gérer 65536 adresses différentes, ou 64 ko de mémoire adressable.

- Lorsque vous installez le circuit de mémoire étendue à 256 ko, le 6502 ne peut pas les "voir" en un seul bloc, car il est limité à 64 ko. C'est comme un immeuble de 256 étages sans ascenseur ni escalier à partir du 65ème palier. Quand une extension mémoire est installée dans un ordinateur, il faut donner à ce dernier la possibilité d'accéder à de petits morceaux de cette extension, appelés "banques", qui vont temporairement remplacer un segment similaire de la mémoire centrale. La commutation de banques est ce que nous utilisons quand nous avons un 130 XE de 128 ko ou un 800 XL de 256 ko. Le 130 XE représente le standard de commutation de banques installé dans tous les XL/XE. Dans le 130 XE il y a une RAM supplémentaire de 64 ko accessible en quatre banques de 16 ko. Tous les autres XL/XE couramment disponibles utilisent cette méthode; c'est la plus grande ressemblance entre toutes ces machines.

LE CONTROLE DE LA MEMOIRE

Vous contrôlez la mémoire étendue en installant ou vidant les bits à l'adresse 54017 décimal ($D301 Hexa), le registre PORT B de la puce PIA. Les huit bits du PORT B sont assignés comme suit:

BIT	FONCTION
7	Contrôle la ROM de 2 ko du diagnostic d'auto-test.
6	Couramment utilisé sur le 130 XE, (contrôle le jeu Missile-Command installé dans le XE GS).
5	Contrôle le processeur vidéo ANTIC pendant le "banking".
4	
3	Bit 1 d'adresse de banque.
2	Bit 0 d'adresse de banque.
1	Contrôle la ROM de 8 ko du BASIC interne
0	Contrôle la ROM de 16 ko du système d'exploitation.

Dans le 130 XE de 128 ko de mémoire, les quatre banques sont adressées de banque 0 à banque 3, ce qui demande que deux bits soient initialisés suivant un de ces quatre modèles: 00, 01, 10 et 11. Pour accéder à une nouvelle banque, positionnez le bit 2 et 3 pour la banque désirée, nettoyez le bit 4 correspondant au 6502, et/ou le bit 5 correspondant à l'ANTIC. Notre nouvelle banque va alors apparaître à la place du second segment de 16 ko de la mémoire centrale à l'adresse allant de 16364 à 32767 décimal ($4000 à $7FFF). Cette surface est appelée fenêtre de banque.

- Avec deux bits vous pouvez adresser quatre banques de 64 ko. Un autre bit permet d'adresser une autre portion de 64 ko (8 banques), pour un total de 128 ko supplémentaires. Un dernier bit adressera 128 ko de plus, (16 banques), pour un total de 256 ko, plus la RAM centrale, ce qui équivaut à 320 ko.

- Voilà le principe utilisé par toutes les extensions mémoire XL/XE. Toutefois, comme il n'y a qu'un seul bit de libre à $D301, nous arrivons à un endroit assez embrouillé car toutes les extensions ne sélectionnent pas le même bit final de contrôle.

- Il y a avantage et inconvénient dans le choix de ce dernier bit. Si vous utilisez le bit 7, l'auto-test est mis hors service. Choisir le bit 1 élimine l'accès au BASIC interne. Ceci n'a pas de conséquence si vous utilisez un langage en cartouche, BASIC ou autre; ceci dit, la plupart des extensions utilisent les deux bits.

TESTONS LES BANQUES

Pour utiliser la mémoire étendue, vous devez déterminer quels bits seront utilisés, et la manière de les adresser. Le but de cette façon de procéder est de déterminer l'ordre dans lequel vous testez. Vous pouvez utiliser une méthode "brute", et tester chaque banque. Après tout, à la vitesse du langage machine, vous n'attendriez pas longtemps; mais nous pouvons le faire de façon plus logique, avec beaucoup moins d'efforts. Examinons comment les octets travaillent ensemble, nous trouverons que peu de tests sont nécessaires en fait. Par exemple, pour tester 192 ko, seul le bit 6 doit être examiné.

- Ce bit suffit à présumer du bon état de la RAM étendue; à cet endroit il y a aussi les banques des bits 2 et 3. Une procédure, pour ce test en assembleur 6502 est de charger d'abord une adresse de banque 130 XE dans le registre Y, et un test d'adresse de banque pour le bit 6 dans le registre X. Ensuite, permutez les banques pendant la comparaison de données à l'intérieur de la fenêtre de banque.

Par exemple:

```
LDY  #$E3      11100011  XE
               BANQUE 0

LDX  #$A3      10100011  Bit 6
               BANQUE 0

STY  PORTB     Autorise XE
               BANQUE 0

LDA  $4000     Prend premier dans la
               fenêtre

STY  PORTB     Autorise Test
               BANQUE 0

CMP  $4000     Autre chose ici?

BNE  FOUND     Oui! Branchement

STY  PORTB     Seulement si la
          donnée est la même

EOR  #$FF      Inverse l'octet

STA  $4000     Ecrit-le en
          retour (l'octet)

STY  PORTB     Et teste encore

CMP  $4000     Toujours pareil?

FOUND....      ZFLAG CLEAR=passe
```

NB: Nous n'avions pas besoin de tester toute la banque 130 XE. Nous l'avons utilisée pour comparer avec notre banque et voir si elles étaient différentes. Comme chaque système de RAM supplétive au bit 6 aura aussi la banque 130 XE, nous pouvons confirmer huit banques avec un seul test.

- SmartRAM 2.5 peut adresser de 64 à 192 ko de mémoire supplétive, donc j'ai démarré en testant les banques 130 XE; si elles passent, je teste le bit 6. J'installe alors un drapeau sur l'état du bit 6 et je teste le bit 5.

- La raison d'être du drapeau du bit 6 est que l'extension 256 ko "NEWELL", en raison de son mode d'adressage particulier, échoue au test du bit 6, quand bien même elle utilise ce bit 6! Si le bit 5 échoue aussi, nous sortons et nous initialisons un RAMdisk de 64 ko; mais si le bit 5 passe et que le 6 échoue, la seule possibilité est que nous avons affaire à une extension NEWELL, alors nous sortons et initialisons un RAMdisk NEWELL.

- Toutefois, si les bits 5 et 6 passent tous les deux, alors nous avons une extension RAMBO-XL compatible avec les machines 256 ko (ou 320 ko). Pour finir, si le bit 5 échoue, nous testons le bit 7.

- Comme vous le voyez, avec seulement 4 bits (XE/6/5/7), nous pouvons vérifier jusqu'à douze banques avec cinq combinaisons différentes de bits et trois formats de mémoire. Maintenant que nous savons tout cela, nous pouvons installer une "table de contrôle de banque", le secret de Smart-RAM 2.5.

TABLE DE CONTROLE DE BANQUES:

La méthode habituelle du choix des bits d'adresse pour le PORT B consiste à utiliser les instructions "bit-shifting" du 6502 afin de les déplacer sur leurs positions. Cela ne fonctionne pas avec SmartRAM 2.5 car nous avons besoin de plusieurs instructions en fonction du bit que vous aurez "shifté" et de sa position. Ma solution a été de considérer le nombre de modèles de bit nécessaire pour chaque combinaison; ensuite je les ai installés dans une "table de contrôle de banques". Il s'agit d'une table d'octets dans laquelle SmartRAM va puiser le bit d'adressage adéquat.

Modèle de table de contrôle

```
130 XE        $E3, $E7, $EB, $EF
Bit 6         $A3, $A7, $AB, $AF
Bit 5         $C3, $C7, $CB, $CF
NEWELL        $83, $87, $8B, $8F
Bit 7         $63, $67, $6B, $6F;
      Banques 4-7
Bit 7Z        $23, $27, $2B, $2F;
      Banques 6-3
Bit 1         $E1, $E5, $E9, $ED
Bit 1Z        $A1, $A3, $A9, $AD
```

- Il y a un total de 32 bits dans la table maître, mais quatre seulement sont utilisés pour 128 ko, huit pour 132 ko, douze pour 256 ko et seize pour 320 ko. A chaque fois que le test passe un niveau, les bits correspondants sont positionnés dans la table de contrôle. Quand le test est fini, la table est lue en entier. Après cela, SmartRAM n'a plus qu'à utiliser une simple division pour déterminer quelle banque est accessible et puiser le bit dans la table. Par exemple il y a 128 "secteurs" simple densité dans chaque banque. Si SmartRAM a besoin du secteur 129, il veut la deuxième banque, donc il prend simplement le second bit dans la table et le pousse dans le PORT B, la banque est alors dans la fenêtre.

L'émulateur 1050

Jetons maintenant un regard au nouvel émulateur 1050 de SmartRAM 2.5. Il tourne sur le 130 XE ordinaire de 128 ko aussi bien qu'avec 192 ou 256 ko. L'émulateur peut gérer les bits 2, 3, 5, 6 et 7 (désolé, mais pas le bit 1). Dans un 130 XE standard de 128 ko, il vous donnera le RAMdisk densité 64 Ko et dans un 256 ko, il vous donnera le RAMdisk "1050" (médium), sans utiliser une seule des banques 130 XE.

- Ceci vous permettra alors de réserver celles-ci à d'autres usages (tel le **BASIC XE**). Une des caractéristiques les plus agréables de Smart-RAM 2.5, est qu'il n'actionne le programme MEM.SAV que si le BASIC (ou une cartouche) est actif. Je suis certain que vous avez déjà été ennuyé par le message **"Type 'Y' if OK use program area"**, alors que vous n'utilisez que le DOS. Ce n'est plus un problème maintenant; si vous n'utilisez pas le BASIC, aucun message MEM.SAV ne s'affichera.

Vous pourrez toujours utiliser la fonction N du DOS (create MEM.SAV) si vous désirez préserver la partie basse de la mémoire. Une autre caractéristique agréable est la possibilité de rebooter sans reformatage du RAMdisk actif. Si vous avez des données dans le RAMdisk, un RESET logiciel $E477 ne les effacera pas. SmartRAM peut être rendu résistant au REBOOT en utilisant le bouton RAMaid de démarrage à chaud d'Innovative Concepts.

Pas de perte de mémoire

- Tout ceci ne vous coûtera absolument rien en mémoire vive car j'ai fait en sorte d'installer SmartRAM à l'endroit utilisé par l'ancien handler Atari; et il ne vous en coûtera aucun espace disque non plus. SmartRAM utilise huit secteurs de longueur, donc un de moins que l'officiel RAMDISK.COM d'Atari.

- Ses limites sont celles propres à tous les disques virtuels, à savoir la perte des données en cas de coupure de l'alimentation et la nécessité d'une sauvegarde sur disquette physique. D'autre part, j'ai choisi de le rendre compatible avec le RAMdisk du DOS 2.5, donc il ne fonctionne qu'avec ce DOS là. Il est incompatible avec les logiciels qui entrent en conflit avec l'ancien handler Atari.

- Quand SmartRAM est chargé dans une machine de plus de 128 ko, il pousse DOS 2.5 à accéder à la mémoire étendue. Si un DOS modifié est opérationnel et qu'il doit être écrit sur un disque, il ne sera pas compatible avec le RAMDISK.COM standard et un 130 XE de 128 ko. Aussi il vaut mieux rebooter sans SmartRAM avant d'écrire les fichiers système sur disquette (option H du DOS). D'ailleurs le menu DOS est modifié pour vous rappeler que votre DOS est bidouillé. Un message est affiché en haut de l'écran.

Deux versions de SmartRAM sont fournies::

SMARTRAM.BAS qui crée un RAMDISK.COM et qui correspond à cet article. Le RAMDISK.COM de 8 secteurs fournit en face 1 du disk CN 12 est une version de SMARTRAM modifiée par MM. Poisson et qui ne charge pas le DUP.SYS ni la MEM.SAV en D8:

Les disques du Cenacle News No 12

Par Rémi Gallopin

Un disk 5,25 pouces est fournit avec ce Cenacle News. La face 1 contient le DOS 2.5 et un RAMDISK.COM de 8 secteurs (voir ci-contre).

Le programme UNARC (qui est ici en AUTORUN.SYS) sert à decompacter les fichiers *.ARC. Avant de l'utiliser, formatez un disk: une face en médium densité ou les deux en simple densité. Puis bootez le disk CN12 face 1 en appuyant sur OPTION. Un menu va s'afficher. Tapez P pour aller dans la fenêtre des options en haut. Puis allez sur "Password Encrypt" et faites <RETURN> pour afficher YES - Si vous utilisez un système à un drive, validez aussi "Disk Swaps".

Nota: si vous avez un 130 XE avec un seul drive, vous gagnerez du temps en recopiant le fichier source (*.ARC) à décompacter dans le D8: et en prenant le D1: comme *destination*.

Tapez ensuite A pour commencer le décompactage, et entrez le nom du fichier *source* (ex: **D8:BASIC.ARC**) puis le numéro du drive *destination* (ex: **1**). Puis entrez le mot de passe qui est **12101893** pour tous les fichiers *.ARC de ce Cenacle News No 12.

Directory de la face 1:

BINARY.ARC
- DEBUG.COM
- ITB32Q.COM & TXT

BASIC.ARC
- DLIMAKE.BAS
- DLIX1.BAS à DLIX3.BAS
- SMARTRAM.BAS
- DISKMEND.BAS
- CYCLOID.BAS & TXT
- TRANSAM.BAS
- TRANSFERT.BAS

UTIL.ARC
- VNTEDTOR.BAS
- HELPER.BAS
- FONCTION.BAS
- MENU.BAS

La face 2 n'est pas compactée et contient les jeux d'action décrits en page 30. Bootez cette face en appuyant sur OPTION, et choisissez un titre dans le menu.

DEBUG+

Par Bryan Schappel

Article paru dans ANALOG COMPUTING de février 1986
Traduit de l'américain par Daniel CARRODANO
Mis en page par Rémi GALLOPIN

Debug+ est un utilitaire de déverminage de langage machine orienté écran. Il contient un programme traceur qui peut circuler à travers pratiquement tout programme en langage machine de trois manières différentes. **Debug+** a aussi un désassembleur défilant complet et un "dumper" de mémoire. Il permet l'exécution du programme de l'utilisateur et peut réaliser des LOADs et des SAVE binaires, plus beaucoup, beaucoup d'autres fonctions. Cela paraît trop joli pour être vrai de la part d'un programme de magazine? Eh bien non, regardez!

Le programme binaire DEBUG.COM est fournit sur le disque du CN 12. Pour le charger depuis le DOS 2.0S Atari, allez au DOS et tapez: **L**

DEBUG.COM

Debug+ sera automatiquement chargé et lancé.

*Pour ceux interressés par le langage assembleur, le code source Mac/65 OSS de **Debug+** était disponible sur le BBS ANALOG Computing TCS et peut être demandé au Club.*

N'oubliez pas, vous devez retirer le **BASIC** pour que **Debug+** se charge. Pour les XL/XE cela signifie de tenir le bouton OPTION appuyé à la mise en route de l'ordinateur.

Le programme réside dans le port cartouche A, aux emplacements mémoire ($A100 - $C0FF) et utilise les adresses $BE00 - $C0FF comme mémoire écran. Le programme reconnaît dix-huit commandes, listées ci-dessous:

TOUCHES	FONCTION
*	Jeu d'adresse
D	Inversion affichage
Q	Quitter **DEBUG+** pour le DOS
G	Aller à adresse
T	Pister le programme
P	Ecrire le desassemblage
E	Effacer mémoire
C	Changer 1 octet de RAM
N	Changer valeur registre
R	Afficher les registres
B	Installer/reinstaller Break Point
S	Sauver un fichier binaire
L	Charger un fichier binaire
F	Trouver une chaîne en memoire
H	Affichage haute vitesse
"	Conversion Dec/Hex, Hex/Dec
_	Défilement mémoire vers le haut
=	Défilement mémoire vers le bas

Avant d'explorer les fonctions, sachez quels sont les messages d'annonces utilisés par **Debug+**. Il y en a trois principaux: ***, ? et (opt1,opt2,opt3)?**

Quand * apparaît dans la fenêtre des entrées, **Debug+** attend un nombre. Ce nombre peut être entré soit en hexadécimal, soit en décimal. Pour entrer un nombre décimal, les chiffres du nombre doivent être précédés par un point décimal. Si le nombre n'est pas precédé par ce point, **Debug+** l'interprêtera comme nombre hexadécimal. S'il y a des chiffres "illégaux" dans le nombre, **Debug+** enverra un signal sonore et la commande sera annulée.

A l'apparition du ?, **Debug+** attend une chaîne de caractères, un nom de fichier par exemple, d'un maximum de douze caractères. Quand vous dépassez la limite de douze caractères, les caractères surnuméraires (à l'exception de DELETE, SHIFT_DELETE et RETURN) viendront recouvrir les premiers douze caractères. [Je conçois que ce soit dur à comprendre, essayez vous-mêmes, vous verrez de quoi il s'agit.]

Le message final est (opt1,opt2,opt3)? Quand ceci apparaît dans la fenêtre des entrées, **Debug+** veut que vous pressiez une des touches separées par une virgule entre les parenthèses. Un exemple en est (D,P)? Ceci vous demande l'unité que vous désirez utiliser, disque ou imprimante. D signifiant disque, et ainsi de suite.

FORMATS D'AFFICHAGE

Debug+ a deux modes d'affichage mémoire: désassemblage et recopie de mémoire. Une ligne désassemblée ressemblera à ceci:

```
ADDR  OP  B1  B1  xMNE
```

ou ADDR est l'adresse, OP le code op, B1 et B2 des opérandes et xMNE est le mnémonique. Le x indique la direction d'une instruction relative (instruction de redirection). Si la flêche pointe vers le haut, c'est un branchement vers l'avant; et si la flêche pointe vers le bas, c'est un branchement vers l'arrière.

Une ligne de mémoire recopiée ressemblera à ceci:

```
ADDR  B1  B2  B3  B4  B5  B6  123456
```

ou ADDR est l'adresse, B1-B6 sont les valeurs dans les emplacements, et 1-6 sont les caractères ATASCII de représentation des valeurs.

COMMANDES

Pour utiliser une des commandes de la table parue plus haut, il suffit de presser la touche du clavier qui lui correspond dans la colonne "Touche". Par exemple, pour mettre à jour le registre d'affichage de ligne, il suffit de presser la touche R.

- **Jeu d'adresse (*).** C'est ainsi que vous dites à **Debug+** l'endroit où débute le désassemblage de la mémoire. Après l'entrée de votre adresse, **Debug+** affichera le contenu de l'emplacement spécifié.

- **Inversion affichage (D).** Cette commande basculera l'écran entre désassemblage et recopie de mémoire, et vice-versa.

- **Quitter Debug+ (Q)**. Cette commande vous renvoie au DOS. Pour revenir à **Debug+** depuis le DOS, faites un "RUN at ADDRESS", avec comme adresse A100. Pour faire ceci avec le DOS OSS/A+, tapez RUN A100 et pressez RETURN. Avec le DOS 2.0s ATARI, tapez M, RETURN, ensuite A100 et RETURN.

- **Aller à adresse (G).** A l'entrée de cette commande, les registres du 6502 sont chargés avec le contenu des registres de l'utilisateur, puis l'exécution du programme commence à l'adresse spécifiée. Le programme de l'utilisateur poursuivra son exécution jusqu'à ce qu'il soit arrêté par une instruction BRK du 6502, un des break points, ou en pressant CTRL-ESC (les touches CONTROL et ESCape simultanément). A l'occurence d'une de ces possibilités, votre programme sera interrompu, les registres 6502 seront sauvegardés dans les registres utilisateurs, et **Debug+** reprendra le contrôle. Ensuite l'endroit où le programme a été arrêté sera désassemblé ou recopié à l'écran, et la ligne de registre sera mise à niveau avec le contenu des registres utilisateur.

- **Pister le programme (T)**. Quand ce mode est activé, **Debug+** demandé l'adresse de départ, puis commence l'exécution du code à l'emplacement donné, une instruction à la fois. Il vous sera demandé de préciser la vitesse du pistage dans le prompt (F,S,O)? Si vous pressez F, le pistage commencera aussi vite que possible; la touche S marquera une pause d'un quart de seconde entre chaque instruction, et la touche O ne poursuivra le pistage que pas à pas après chaque appui sur la touche OPTION. Un pistage sera annulé si le pisteur trouve: (1) une instruction BRK 6502, (2) un break point, (3) une instruction illégale, et (4) par l'appui sur la touche ESC. NB: En cas d'utilisation du mode pas à pas, il faudra presser OPTION plus ESC pour obtenir une annulation. Une pause peut être marquée dans le pistage en pressant sur la BARRE d'ESPACE. Le pisteur connaît quelques limites. Tout d'abord il ne peut pas se pister lui-même, n'essayez donc pas de pister **Debug+**. Deuxièmement, chaque tentative de pister les routines E/S en temps réel (comme disque ou cassette) tournera à l'echec. Durant un pistage, la ligne registre utilisateur sera mise à niveau avant et après l'exécution de chaque instruction, de facon que vous puissiez examiner le contenu du registre à tout instant.

- **Ecrire le désassemblage (P)**. Quand cette commande est exécutée, il vous est demandé vers quel périphérique les données doivent être expédiées (disque ou imprimante) et depuis qu'elle adresse de départ. Si la sortie disque est choisie, un nom de fichier vous sera demandé, ensuite le désassemblage commencera. Durant le processus, l'adresse tout juste envoyée au périphérique est affichée sur la ligne de STATus. Pour annuler cette commande lors de son exécution, pressez une touche quelconque. Si vous choisissez le disque comme périphérique de sortie, **Debug+** créera un fichier texte du désassemblage. Vous pourrez ensuite charger celui-ci dans un traitement de textes, ou un éditeur de textes, et éditer le désassemblage.

- **Effacer mémoire (E)**. A l'exécution de cette commande, il vous sera demandé les adresses de début et de fin de l'effacement. Après ceci, la mémoire comprise entre ces deux adresses sera effacee. **Debug+** remplira cette zone de zéros (l'instruction BRK du 6502). Si l'adresse de fin est plus petite que l'adresse de début, vous obtiendrez un "ADDRESS RANGE ERROR", et la commande sera annulée.

- **Changer un octet de RAM (C).** A l'entrée de cette commande, il vous est demandé l'adresse où faire un changement et quoi y changer. A ce moment le contenu de cet emplacement mémoire sera modifié. Ceci est semblable au POKE du Basic. Assurez-vous de ne pas changer une partie de la

mémoire utilisée par **Debug+**. De même, le manque de précaution dans le choix d'une adresse peut effacer une donnée vitale du systeme et provoquer un plantage général. Faites attention.

- **Changer valeur registre (N)**. Cette commande vous demandera quel registre modifier. Pour choisir le registre, il suffit de presser une des touches suivantes: A pour Accumulateur, X pour Registre-X, Y pour registre-Y, S pour Stack Pointer (Pointeur de pile), ou P pour Status du Processeur. Après, il vous sera demandé la nouvelle valeur. Entrez celle-ci et le contenu du registre sera changé. Notez que la ligne de registre en haut de l'écran a été modifiée de manière à refleter la nouvelle valeur. Une note au sujet de la ligne de registre. Si vous examinez la ligne a l'écran, vous verrez cinq petites boites et une grande, étiquetée NV_BDIZC. Dessous l'étiquette, il y aura huit chiffres binaires. Ceci est l'état du processeur, tel que le montre l'analyse de ses drapeaux. La table suivante vous montre de quels drapeaux il s'agit:

N	Drapeau de nombre négatif
V	Drapeau de trop plein
_	Non utilisé (toujours affiché ainsi)
B	Drapeau d'instruction BRK
D	Drapeau décimal
I	Drapeau interruption.
Z	Drapeau zéro
C	Drapeau déplacement (Carry)

- Si un 1 apparaît dessous un de ces drapeaux, il indique une condition "installée" où "vraie". Donc, si le bit I est 1, c'est qu'une interruption se déroule.

- **Registres d'affichage (R).** Cette commande ne fait rien d'autre que mettre a niveau la ligne de registre. Un point d'arrêt arrêtera l'exécution (ou le pistage) d'un programme utilisateur. Quand vous entrez cette commande, un B apparaît dans la fenêtre des entrées, indiquant que vous pouvez soit mettre soit retirer un point d'arrêt. Il est facile d'installer un point break. Il suffit de taper le numéro de point d'arrêt (un simple chiffre entre 1 et 6), suivi d'une virgule, puis l'adresse à laquelle vous désirez mettre le point d'arrêt. Voici un exemple typique: B1,0600 [RETURN]. Ceci indique à **Debug+** d'installer le point d'arrêt numéro 1 à l'emplacement $0600 (ou 1536 décimal). Si le point d'arrêt numéro 1 est deja initialisé, vous aurez un message d'erreur. Si l'emplacement $0600 contient une instruction BRK, vous aurez aussi une erreur. Toutes les instructions BRK 6502 sont considérées comme des points d'arrêt. Quand vous désassemblez l'adresse où un point d'arrêt est positionné, le mnémonique sera indiqué en inverse vidéo. C'est ainsi que **Debug+** montre où se trouvent les points d'arrêt, de facon graphique. Si vous êtes dans le mode de recopie de mémoire, la valeur de l'adresse où se trouve le point d'arrêt sera elle aussi affichée en inverse vidéo. Retirer un point d'arrêt est encore plus facile que de le mettre. Au prompt B, il suffit d'indiquer le numéro du point d'arrêt que vous désirez supprimer, suivi de [RETURN]. Par exemple, B1[RETURN]. Cela va remettre à zéro le point d'arrêt numéro 1. Si celui-ci n'etait pas installé, vous aurez une erreur. Quand un point d'arrêt est supprimé, l'ancien opcode est restauré à l'adresse du point d'arrêt, et l'adresse du point d'arrêt est remise à $0000. Si vous regardez la ligne du point d'arrêt en bas de l'écran quand vous chargez **Debug+**, vous noterez que les adresses du point d'arrêt sont 0 et qu'il y a six points d'arrêt libres.

- **Sauvegarder un fichier binaire (S).** Cette commande vous permet de sauvegarder sur disque un programme binaire monobloc. Il vous sera demandé d'entrer le nom de fichier, suivi des adresses de départ et de fin du bloc. Une fois encore, si l'adresse de fin est inférieure à l'adresse de début, vous aurez une erreur. Le fichier créé par cette fonction peut être chargé depuis le DOS. Vous ne pouvez

pas préciser une adresse de run pour le fichier avec **Debug+**, donc vous pouvez avoir à le faire depuis le BASIC ou le DOS.

- **Charger un fichier binaire (L).** Ceci va charger tout fichier binaire en mémoire. **Debug+** ne lancera pas ce fichier. Après que vous aurez entré le nom du fichier à charger, **Debug+** chargera celui-ci en mémoire, puis il vous affichera l'adresse de chargement initiale dans la ligne STATus.

- **Trouver une chaîne en mémoire (F).** Cette commande localisera toutes les occurences d'une chaîne - jusqu'à douze caractères de long - dans la mémoire. Il vous sera demandé d'entrer la chaîne à rechercher et de presser RETURN. **Debug+** effacera l'écran, et chaque occurence de ce que vous avez indiqué sera affichée à l'écran, dans ce format:

FIND # nn Hexadr Decadr

où nn est le numéro d'occurence et Hexadr et Decadr sont les adresses hexadécimale et décimale de l'occurence. La recherche se poursuivra jusqu'à ce que la fin de la mémoire soit atteinte ($FFFF ou 65535 décimal). Vous pouvez trouver utile de commencer la recherche à une adresse donnée. Pour ce faire, terminez votre chaîne avec une virgule. **Debug+** vous demandera depuis qu'elle adresse débuter, et la recherche commencera comme décrit ci- dessus. **Debug+** possède certains caractères "réservés" qui ne peuvent pas être insérés dans vos entrées, et ne peuvent par conséquent pas être recherchés. Ce sont: (1) RETURN (ATASCII 155), (2) DELETE (ATASCII 126), (3) SHIFT-DELETE ou DELETE LINE (ATASCII 156), et (4) ESC (ATASCII 27). Tous ces caractères sont utilisés par **Debug+** soit comme délimiteurs, soit comme caractères de contrôle. Toutefois tous les autres caractères contrôle et alpha-numériques sont à votre disposition. Une autre chose, si vous désirez rechercher un caractère que vous générez habituellement en pressant ESC, comme la commande vide écran (ESC-CTRL-CLEAR), alors entrez la touche sans presser ESC. Donc, en ce qui concerne le vidage d'écran vous entrerez simplement CTRL-CLEAR.

Affichage haute vitesse (H). Cette commande fera défiler en continu à l'écran le contenu de la mémoire. Il vous est demandé le sens du défilement - vers le haut ou vers le bas - avant de le faire débuter. Utilisez cette commande pour aller très rapidement quelque part, puisque l'écran va se modifier sans que vous puissiez y voir grand chose. Pour marquer une pause dans le défilement, pressez ESPACE; pour annuler le défilement pressez ESC.

Conversions Decimal/Hexa et Hexa/Decimal (.). Cette commande va convertir une base numérique en une autre et afficher les résultats dans la ligne de STATus. Si un nombre hexadécimal est entré, un nombre décimal sera généré, et vice-versa. Après que le nombre à convertir a été entré, la ligne STATus ressemblera à ceci: nnnn = xxxx, où nnnn est ce que vous avez tapé, et xxxx en est la forme convertie.

Défilement vers le haut de la mémoire (-). Quand vous pressez la touche tiret, **Debug+** va remonter la fenêtre d'affichage d'un octet dans la mémoire. Si vous êtes dans le mode de recopie de mémoire, **Debug+** remontera la fenêtre de 6 octets. Vous pouvez remarquer qu'il faudra normalement plusieurs actions sur les touches pour faire défiler une expression dans la mémoire. Ceci parce que **Debug+** n'a pas la moindre idée de l'endroit où débute l'instruction précédente, ni de sa longueur en octets. A l'exécution de cette commande, les adresses seront plus basses.

Défilement vers le bas de la mémoire (=). Cette commande fera reculer la fenêtre d'affichage d'une instruction entière dans la mémoire (ou de 6 octets en mode de recopie). Durant le défilement vers le bas, **Debug+** connait l'endroit où la prochaine instruction débute. Voila pourquoi il peut reculer d'une instruction. A l'utilisation de cette commande, les adresses dans la fenêtre d'affichage seront plus hautes.

NOTES

(1) Pour chacune des commandes relatées ci-dessus, l'auto-répétition du clavier est active. Aussi vous pouvez renouveler chaque commande en tenant sa touche enfoncée, aussi longtemps que necessaire. C'est particulièrement utile dans l'utilisation des commandes de défilement.

(2) Durant l'exécution de chaque commande, l'appui sur la touche ESC provoquera une annulation. Si vous êtes en train d'entrer un nom de fichier et que vous décidez de ne pas charger ce fichier, appuyez à tout moment sur ESC pour annuler l'action.

(3) A tout moment, pour arrêter le défilement vous pouvez appuyer sur ESPACE. Pour annuler une commande durant laquelle l'écran défile sans arrêt, appuyez sur ESC.

NE FAITES PAS.

Bon, on a vu ce que vous pouvez faire avec **Debug+**, je vais maintenant vousentretenir de ce que vous ne pouvez pas faire.

(1) Ne modifiez aucunement le contenu de la mémoire située entre $A100 et $C0FF.c'est là que réside **Debug+**. Le changement d'une partie de cette mémoire seraitfatal.

(2) **Debug+** utilise le vecteur VBREAK situé à l'adresse $0206-$0207 pourdétecter l'instruction BRK du 6502. Ne modifiez pas ce vecteur. Ce faisant vous abimeriez sévèrement **Debug+**. A votre sortie de **Debug+**, le programme restaurera le vecteur VBREAK par defaut.

(3) **Debug+** utilise aussi le vecteur VKEYBD situé à l'adresse $0208-$0209 pour détecter la combinaison de touche CTRL-ESC durant un run par l'utilisateur. Ne modifiez pas ce vecteur. Si vous le faites vous ne pourrez plus jamais arrêter l'exécution d'un de vos programmes. A la sortie de **Debug+**, le programme restaure le vecteur par defaut.

(4) Quand vous pressez sur RESET à l'intérieur de **Debug+**, le programme prendra le contrôle et se réinitialisera lui-même. Ne modifiez pas le vecteur DOSINI situé à l'adresse $0C-$0D. A votre sortie de **Debug+**, le vecteur DOSINI est restauré.

Eh bien voila. **Debug+** tournera comme indiqué, à moins que vous n'outrepassiez une des règles que j'ai soulignées ci-dessus. Il vous sera d'une aide appréciable dans le déverminage de vos programmes en langage machine.

PROGRAM HELPER

par Jonathan Stone

ANALOG COMPUTING février 1986

traduit de l'américain par Daniel CARRODANO

Convertit les constantes en variables pour économiser de la RAM.

Avez-vous déjà eu un message de mémoire insuffisante en mettant la touche finale a votre plus beau chef d'oeuvre BASIC? L'épouvantable erreur 164 n'a-t'elle jamais anéanti votre programme? Si oui, et même sinon, jetez un regard sur cet utilitaire: PROGRAM HELPER.

CE QUE FAIT LE PROGRAMME.

- Une des choses que fait HELPER c'est de diminuer le montant de RAM occupé par le programme en convertissant en variables les constantes fréquemment utilisées. Par exemple, si la constante 8 est utilisée dix fois dans un programme, si vous demandez à Helper de convertir ce nombre, il définira la variable C8 comme équivalent de 8 en début de votre programme, et remplacera ensuite chaque 8 par C8. Vous devez garder la ligne 0 disponible dans votre programme si vous voulez utiliser cette option.

- Du fait qu'une constante occupe 7 octets dans un programme alors qu'une variable n'en occupe qu'un seul, ceci récupérera un total de 60 octets (moins les octets utilisés dans la ligne 0). Mais ce n'est qu'un début. Chaque constante qui apparaît plus de deux fois vaut la peine d'être convertie en variable. Et, dans un grand programme, quelques constantes peuvent se manifester plusieurs centaines de fois.

- Si Helper est lancé sur lui-même, par exemple, la conversion des constantes 0-10, 20, 34, 100, et 255 diminuera la RAM occupée de plus de 15 pour cent. Vous pouvez trouver les constantes les plus utilisées dans votre programme simplement en les examinant avec LIST. - Helper vous autorise aussi à changer le nom de toute variable dans votre programme. A la demande, il suffit d'entrer le nom de la variable à changer, puis le nouveau nom, et c'est fait. Assurez-vous simplement, si le nom de la variable se termine par (ou par $, que le nouveau nom ait la même terminaison. Ces terminaisons indiquent au BASIC si la variable est une chaîne ou un tableau.

- Il y a un autre avantage dans Helper. Si votre programme contient une erreur 164, il est en principe irrécupérable sans une connaissance particulière de la structure du DOS. Ceci parce-que le BASIC efface ce qui a été préalablement chargé avant l'erreur - ne vous laissant rien. Si vous lancez Helper sur un tel fichier, le fichier de sortie recouvrera tout jusqu'à la survenance de l'erreur, et ceci peut être entré (par ENTER) dans la mémoire. Même si une partie du programme est malgré tout perdue, j'ai plus d'une fois trouvé cette option extrêmement utile.

COMMENT L'UTILISER

- Quand c'est demandé, donnez le nom du programme que vous voulez modifier ainsi que le nom sous lequel sera sauvegardé ce fichier une fois qu'il aura été modifié. Pour un répertoire de disque pressez RETURN à la première demande. Si vous voulez modifier un nom de variable, ou convertir une constante en variable, pressez simplement Y lors de la demande adéquate. Le nombre maximum alloué pour la conversion de constantes est limité à 25. Une fois que Helper a commencé la création du fichier de sortie, vous pouvez quitter à la fin de chaque ligne en pressant ESC.

COMMENT IL MARCHE

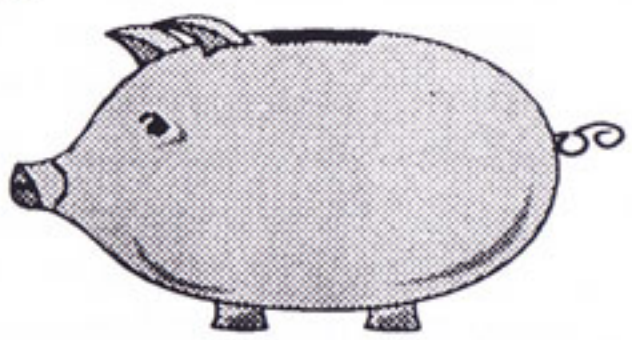
- Helper prend en aparté un fichier sauvegardé (ou compacté), octet par octet, et le transforme momentanément en programme listé (NdT-sous forme ASCII), comme le fait le BASIC quand un fichier est chargé. (Si vous n'êtes pas intéressé par le moyen dont ceci est réalisé, sautez cette partie).

- Pour comprendre comment le programme travaille, vous avez besoin de quelques informations sur la structure d'un fichier BASIC compacté. Pour des informations détaillées sur la structure d'un programme BASIC, lisez le livre THE ATARI BASIC SOURCE BOOK ou DE RE ATARI.

- Un fichier sauvegardé commencera toujours par sept pointeurs de 2 octets. Les deux premiers octets sont toujours à zéro. Helper utilise les six autres pointeurs pour trouver les différentes tables du fichier. Ce sont: la table des noms de variable (Variable Name Table), la table des valeurs de variable (Variable Value Table) et la table des formulations (Statement Table). - La table des noms de variables est simplement une liste de toutes les variables utilisées dans un programme. Dans Helper, la table tout entière est mise dans une chaîne. La table de valeur de variable est l'endroit où l'espace est réservé pour les valeurs des variables. Helper saute à cette table.

- La table des formulations est le corps du programme. Elle contient, sous forme compactée, toutes les lignes du programme en ordre numérique. Chaque ligne consiste en un nombre de lignes sur 2 octets, un symbole de fin de ligne et une ou plusieurs formulations.

- Chaque formulation est composée d'un symbole de longueur de formulation, un symbole de commande et un nombre variable d'autres symboles. (La commande est le premier symbole exécutable dans une formulation et détermine comment les symboles suivants seront interprétés). Un symbole suivant la commande sera d'un des types suivants: variable, constante, opérateur, fonction, double guillemets, autre formulation, ou un "end-of-line" (EOL) (FIN DE LIGNE). Chaque type possède une routine qui le gère (voir description du programme).



- Helper prend chaque symbole dans la table de formulations depuis le disque et affiche la liste des formulations à l'écran, aussi bien que sur le disque. Ainsi, vous pouvez voir le programme durant sa création. Après que le programme listé aura été terminé, il sera ensuite rechargé automatiquement par ENTER et Sauvegardé par SAVE - recréant ainsi le fichier de sortie Basic. NDLR: j'insiste sur le fait que tout se déroule automatiquement!

DESCRIPTION DU PROGRAMME

Lignes	Fonction
10-70	Initialise écran et lecture des tables formulation et opérateur/fonction
80-160	Entrée fichier à modifier et sortie fichier.
170-290	Charge table noms de variables depuis fichier et la met dans VAR$. La table de valeur de variable est sautée.
300-350	Obtient la longueur de ligne, lon-gueur de formulation et une commande 1-octet.
360-390	Gèrent les commandes
400-410	Gérent les formulations REM et d'erreur. Le contenu actuel de ces formulations est ignoré. Seule la commande est envoyée au fichier de sortie
420-490	Obtient les commandes compactées et détermine s'il s'agit d'un des cas spéciaux (autre qu'opérateur ou fonction) nécessitant des routines séparées.
500-540	Gère opérateurs et fonctions.
550-600	Gère les variables.
610-690	Gère les constantes. La ligne 670 convertit les constantes en variables.
700-730	Gère les chaînes.
740-760	Gère les données de formulations.
770-790	Aide pour l'utilisation des formulations THEN et le symbole EOL.
800-850	Routine fin de fichier "End-of-File".
860-950	Diverses routines de chaînes utilisées dans le programme.
960-990	Données Table commandes.
1000-1040	Données Table Opérateur/Fonction.
1040-1190	Routine vous permetant de changer les noms de variables.
1200-1330	Entre quelles constantes sont à convertir en variables.

NDLR: Tout ce processus peut durer un certain temps, donc soyez patient, ou alors travaillez sur RAMdisk pour aller plus vite! Au début vous verrez le programme s'afficher à l'écran, mais après, pendant le **Enter** et le **Save**, il ne se passera plus rien pendant quelques minutes, surtout si vous travaillez sur Ramdisk, donc attendez le retour du message **Ready** avant d'agir inconsidérément (c'est-à-dire ne faites pas Break ni RESET!).

Maintenant, laissez **PROGRAM HELPER** vous aider à créer vos chefs-d'oeuvre en BASIC.

PROGRAM HELPER figure sous le nom HELPER.BAS dans l'ARChive UTIL.ARC

VARIABLE NAME TABLE EDITOR

par Earl Davidson, A.N.A.L.O.G. février 1986
Traduit de l'anglais par Daniel CARRODANO

Editeur des noms de la table des variables

J'ai tendance à modifier ma façon de penser au sujet des noms de variables pendant que je programme. L'ennui, c'est que de changer le nom d'une variable mainte fois utilisée à l'intérieur d'un programme est fastidieux, vu que le BASIC ne fournit pas de commandes pour le faire, ni de fonction recherche et remplacement. J'ai donc écrit ce programme pour me faciliter la tâche. Il lit l'en-tête de la table de noms de variables (VNT) dans un programme sauvegardé par SAVE, autorise la modification de chaque variable, puis écrit un nouveau programme sur disque.

Lors de l'entrée d'un programme BASIC, chaque ligne de texte que l'on tape avec l'éditeur du BASIC passe par un processus que l'on nomme pré compilation. Le BASIC convertit chaque ligne dans une forme compactée ("tokenized") qui sera interprétée beaucoup plus vite durant l'exécution du programme. La version compactée de la ligne est placée dans la table des formulations "Statement Table" (ST). D'autres tables sont établies durant le stade de pré compilation: La table des noms de variables "Variable Name Table" (VNT), la table des valeurs de variable "Variable Value Table" (VVT), et la table des chaînes et aires"String/Array".

Si la ligne entrée contient une nouvelle variable, son nom est ajouté à la VNT, et une information au sujet de la variable est placée dans la VVT. Si la variable est une chaîne ou un tableau, elle est ajoutée à la table String/Array. Dans la version compactée d'une ligne de programme, chaque variable est représentée par un symbole ("token") d'un octet. La valeur de cet octet est la position de la variable dans la VNT, augmentée de 128 ($80). Un programme BASIC ne peut avoir que 128 variables, et leurs symboles vont de 128 à 255 ($80 à $FF).

La VNT est maintenue par le BASIC à la convenance du programmeur. Elle est utilisée seulement par la formulation LIST. Quand le programme, ou une portion du programme, est listé, le symbole de la variable est utilisé comme index dans la VNT et le nom de la variable est affiché. Le nom de la variable peut avoir n'importe qu'elle longueur. Il doit consister en lettres majuscules et nombres et doit commencer par une lettre. Ceci nous donne une grande liberté dans l'utilisation de noms de variables significatifs de manière que notre programme soit facile à relire.

Quand un programme est écrit sur disque en utilisant la commande LIST, le fichier est exactement le même que lorsque le programme est listé à l'écran. Si ensuite il est entré par la commande ENTER, le BASIC précompile chaque ligne et installe les différentes tables tout comme si cela était fait au fur et à mesure.

Toutefois, quand un programme est sauvegardé sur disque par la commande SAVE, le fichier consiste en un en-tête de 14 octets et les diverses tables établies. La ST est sauvegardée dans son format compacté. Quand le fichier est par la suite rentré avec la commande LOAD, l'en-tête du fichier est utilisé par le système d'exploitation comme un index des différentes tables et celles-ci sont simplement lues et placées en RAM à l'emplacement adéquat.

Quand nous entrons un programme, si nous changeons un nom de variable, le nouveau nom est placé dans la VNT. Le nom original reste dans la table, malgré qu'il ne soit plus utilisé par le programme. Le seul moyen d'effacer le nom original et de récupérer de la RAM et de la place pour les variables est de LISTer le programme sur disque, taper NEW, et rentrer à nouveau le programme (avec ENTER) en mémoire, ce qui va créer une nouvelle VNT n'utilisant que les noms de variables rencontrés dans le fichier.

UTILISATION DE VNT EDITOR

Quand VNT Editor tourne, un menu de quatre items est présenté: Read file, Write file, Edit Variables Names, et Change All. Vous devez d'abord lire le fichier que vous désirez modifier. Durant l'édition des noms de variables, vous devez prendre garde d'utiliser des noms valables si vous voulez pouvoir ensuite éditer le programme. Bien que le BASIC ne fasse aucune restriction à la longueur des noms de variables, VNT Editor vous limite à 30 caractères par nom, ce qui devrait suffire. Remplacer un nom court par un autre plus long peut aussi être un problème s'il en résulte des lignes de programme plus longues que l'éditeur du BASIC ne les accepte. Quand vous avez terminé vos changements, sauvegardez le fichier sous un nom différent. Le nouveau fichier peut alors être chargé par un LOAD).

VNT Editor ne fait aucun contrôle sur la légalité des noms de variables ni sur les doublons éventuels. Il peut y avoir des occasions où vous désirez ne pas utiliser de noms légaux. Si votre programme utilise toute la RAM et que vous avez besoin de quelques octets, vous pouvez vouloir remplacer tous les noms de variables par des noms en une seule lettre. Le BASIC ne vous en accorde que 26 (A à Z) et considère le signe dollar des chaînes et les parenthèses des aires comme faisant partie du nom de variable. L'option Change All vous permet de changer tous les noms de variable dans le même nom d'un seul caractère, ce qui peut vous économiser plusieurs centaines d'octets dans un grand programme. Si vous avez un programme que vous voulez rendre non modifiable par les autres, pour une raison quelconque, vous pouvez modifier les noms de variables avec des caractères spéciaux. Le fichier résultant tournera correctement, mais si une ligne contenant un nom de variable non valide est éditée il en résultera une erreur qui ne pourra pas être corrigée. Essayez de changer tous les noms de variables par une chaîne contenant 30 points d'interrogation. Le fichier résultant tournera correctement, mais s'il est LISTé il apparaîtra de façon étrange.

Si vous créez un fichier avec des noms de variable illégaux ou dupliqués, prenez soin de garder une copie de votre fichier original en vue utilisation future.

EXPLORATION DU PROGRAMME

VNT Editor est direct, sans astuces ni routines complexes. Les remarques ne sont pas référencées, donc vous pouvez les omettre si vous le désirez. La routine Read File commence à la ligne 810. La ligne 830 lit l'en-tête de 14 octets du fichier. La ligne 840 contrôle les deux premiers octets. Ils doivent être tout deux des zéros décimaux si le fichier est un programme en BASIC sauvegardé. Les troisièmes et quatrièmes octets contiennent un offset (image miroir (depuis zéro)) depuis la fin de l'en-tête jusqu'au commencement de la VNT+256. VNTSTART est initialisé à cette valeur en ligne 850. L'offset à la fin de la VNT est plus petit de un que le début de la VVT. La ligne 860 assigne cette valeur à VNTEND.

Les septièmes et huitièmes octets de l'en-tête contiennent l'offset du début de la VVT+256. VVTSTART est initialisé à cette valeur en ligne 870. Offset de la fin de la VVT est inférieur de 1 au début de la ST. La ligne 880 assigne cette valeur à VVTEND. Les neuvièmes et dixièmes octets contiennent l'offset du début de ST+256. La ligne 890 assigne cette valeur à STSTART. Les treizièmes et quatorzièmes octets contiennent l'offset de la fin de ST+256. La ligne 900 assigne cette valeur à STEND.

Les cinquièmes, sixièmes, onzièmes et douzièmes octets de l'en-tête ne sont pas utilisés par le BASIC ni par VNT Editor.

La VNT, comme déterminée par VNTSTART et VNTEND, est lue et placée dans la chaîne VNT$. La longueur de VNT$ est stockée dans VNTLEN1 pour utilisation ultérieure. Après édition des noms de variable, un nouveau fichier est écrit par la routine commençant à la ligne 1010. La ligne 1040 vérifie que le nom de fichier destination est différent du nom de fichier source. La longueur courante de VNT$ est stockée dans VNTLEN2 en ligne 1060, et la différence entre VNTLEN1 et VNTLEN2 est stockée dans CHANGE à la ligne 1070. Les offsets trouvés dans HEADER$ sont mis à niveau par la valeur de CHANGE dans les lignes 1080 à1160.

La ligne 1180 écrit les HEADER$ dans le nouveau fichier. La ligne 1190 écrit la nouvelle VNT$ sur le nouveau fichier.

(suite page 31)

A PROPOS DU ST XFORMER

Par Daniel Carrodano

Dans le Newsquick No 11, Michel Breton a traité de l'interface ST XFormer et de son logiciel d'émulation XE. Un autre article à ce sujet figure également dans le Cenacle 11.

La première des choses que je désire vous communiquer c'est que le câble XFormer ne s'emboîte pas dans le port série du ST, mais bel et bien dans le port imprimante parallèle (port Centronics) de ce dernier. Ce qui peut induire en erreur c'est l'utilisation sur le ST d'une prise DB 25 pour le port parallèle. Mais cela ne prête pas à conséquence car le câble ne peut pas s'enficher sur la prise série du ST. Par contre, si vous fabriquer le câble vous-mêmes vous devrez en tenir compte.

Si un ST avec 1 Méga de RAM est conseillé car il permet d'utiliser les logiciels de transfert livré avec l'émulateur, est-ce à dire que tout est perdu si vous n'avez que 512 ko? Pas du tout! Je suis dans ce cas, et je vais vous montrer que vous pouvez procéder au transfert de fichier dans les deux sens, très facilement.

Il me fallait transférer des fichiers XE contenus dans une disquette PC 3,5". Sachant que le ST sait lire les disquettes PC, j'avais la conviction qu'on pourrait y arriver grâce au XFormer. Mais aussi, comment des programmes 8-bits peuvent-ils se retrouver dans une disquette PC?

Si vous êtes abonné à ATARI CLASSICS, AC pour les intimes, le nouveau magazine mondial entièrement consacré à nos 8-bits, (les Classics pour les mêmes intimes), vous n'ignorez plus qu'il existe de bonnes versions de langage de programmation "C" pour le Classic, et dans le domaine public qui plus est. Seulement il fallait les télécharger sur le BBS de l'université de l'état du Michigan à 8000 km de chez nous (purée de nous autres!), de l'autre côté de l'océan. Simple comme bonjour si votre Classic est relié au téléphone par un modem. Mais moi, j'ai l'ordinateur, le téléphone, le modem, le logiciel de télécommunication, mais le tout dans le désordre; autrement dit ce n'est pas fonctionnel. C'est dans de telles circonstances qu'on se tourne vers les amis, et c'est ce que j'ai fait. J'ai demandé à plusieurs amis (abondance de biens de nuit pas) car je sais par expérience que les disquettes venant de l'étranger peuvent présenter des défauts (bad sector, CRC, etc). De Jim Cutler, en Angleterre, et d'Alex Pignato, (le président du club des Ol' Hackers dont j'ai l'honneur d'être membre) New York, j'ai reçu la disquette CC8. Qu'ils en soient ici remerciés. J'avais aussi demandé à Michel Savoski, qui n'est pas un Atarien, mais un PCiste, le programme CC65. Michel partage avec moi la passion du Domaine Public... Grâce à lui j'ai eu la surprise de trouver dans ma boite aux lettres, quelques jours plus tard, une disquette PC contenant les fichiers du programme CC65, disquette envoyée de Cologne, en Allemagne par monsieur Dirk Hermann, que je remercie également.

Le monde est petit n'est-ce pas? Il me fallait transférer sur disquette 5,25" des programmes XE originaires des USA, transitants par l'Allemagne, et tout cela en utilisant un ST... Moi j'aime ce genre de situation, pas vous? Pourquoi faire simple quand on peut faire compliqué? Non, ce n'est pas cela du tout. Je dirai plutôt qu'il faut s'avoir utiliser les moyens dont on dispose, en n'ayant pour but que le résultat final. Mais puisque la boucle est bouclée, je referme ma parenthèse.

Sur la disquette PC il y avait des programmes archivés avec l'utilitaire ARC de Bob Puff, qui a été traité dans le numéro 1O de Cenacle News. Pour les désarchiver il fallait d'abord les transférer sur une disquette au format 8-bits. En lisant la documentation du ST Xformer, je vis que cet émulateur possède une caractéristique fort intéressante, en effet le drive interne du ST est reconnu comme périphérique F: dans le mode XE. Le mode XE est l'émulation XE du logiciel XFormer. Cette unité F: ne répond pas comme les autres, pas possible d'en tirer un directory, ni d'y manipuler les fichiers depuis le DOS ou avec des commandes XIO, elle est "orientée octets". J'entends par la qu'on peut y manipuler les octets un par un. C'est

peu direz-vous? Peut-être, mais pour moi c'était amplement suffisant. Un tout petit programme BASIC allait me tirer d'affaire. Le programme XFORMER.BAS est fournis sur le disk CN 12.

Ce programme, bien que sommaire, remplit parfaitement sa fonction, qui est de copier des fichiers de ST à XE et vice-versa. J'ai essayé d'y inclure quelques sécurités, car toute erreur dans la spécification de fichier pour l'unité F: entraîne un plantage du XFormer. Hélas cette sécurité ne joue pas, il faut donc ne pas se tromper. Pour le périphérique en destination, le nom de fichier est laissé à votre convenance, mais là le rôle de la sécurité c'est d'éviter de se tromper d'unité. Le transfert proprement dit utilise les instructions BASIC GET et PUT. Le programme ouvre un canal en lecture et un en écriture. Il pioche un octet dans une unité et le reverse dans l'autre. Mais l'ordinateur ne fait pas ceci pas à pas, en réalité il en stocke un certain nombre dans un tampon et recopie ensuite le tampon d'un seul coup. Malgré cela ce n'est pas vraiment très rapide, surtout que le BASIC XFormer n'atteint que 60% de la vitesse du Basic Atari (d'après la doc). Alors pour accélérer le processus il est préférable d'utiliser le périphérique D8: pour remplacer un des périphérique. On pourrait croire que le D8: n'est pas utilisable avec un ST de 512 ko? Et pourtant si, car le D8: n'est pas un RAMdisk interne du XFormer, et il est géré à part. Pour avoir le D8: il faut booter le mode XE depuis le drive externe D1: (drive XE), en ayant dedans une disquette contenant le DOS 2.5 avec le fichier RAMDISK.COM, d'aller au bureau XFormer avec la touche F5 sur le ST, et de cliquer sur "Power on reset". Cela simule un arrêt suivi d'une remise en route du XE, et c'est justement utilisé pour booter depuis une disquette externe. A l'écran apparaît le message d'installation du D8: suivi soit du prompt READY du BASIC, soit du menu DOS, selon que le BASIC du XFormer est ou non présent. En effet le XFormer comporte la possibilité de supprimer le BASIC du mode XE. Pour accélérer encore le processus, j'ai adjoint au programme la possibilité d'éteindre l'écran. C'est sensé permettre d'aller plus vite. Toutefois, je n'ai procédé à aucun chronométrage. Si le fichier à copier est plus grand que la place libre sur la disquette (ERROR162), vous avez alors le message "PLUS DE PLACE SUR LA DISQUETTE CIBLE", et le morceau de fichier déjà copié est effacé de la disquette. En effet un fichier incomplet ne peut servir à rien, si ce n'est à induire en erreur.

ATTENTION: de F: vers D8 le transfert marche, mais semble ne pas marcher de D8: vers F: !!! C'est à dire que le fichier cible n'est pas au directory ou alors a une taille de zéro octet.

Maintenant je vais vous parler d'un bug du XFormer, (mais c'est peut-être un bug de mon programme!!!); il lui arrive de modifier l'indication de la taille du premier fichier disponible sur la disquette cible, y compris dans le D8:. Par exemple, si le fichier "fait" 100 secteurs de long, le DUP.SYS du D8: va se retrouver avec une indication de 115 secteurs. Ceci même s'il est protégé en écriture. L'indication du nombre de secteurs libres n'est pas erronée, et le fichier n'est pas corrompu, seule l'indication de sa taille a changé. J'ai tourné la difficulté en écrivant toujours en premier les fichiers DOS sur la disquette D1:. Ainsi, si je transfère directement sur D1:, il me suffira ensuite d'écrire à nouveau les fichiers DOS pour corriger l'anomalie.

Dans l'utilisation du XFormer il y a un désagrément notable, le clavier assumé est "américain" (QWERTY), alors que les ST français ont un clavier AZERTY. Ca ne doit pas vraiment être un problème, il existe très certainement un accessoire tournant la difficulté. Même avec un 512 Ko il reste 13 Ko de libre, ça doit être amplement suffisant pour un accessoire de ce genre. Et si quelqu'un possède un tel accessoire du domaine public, je suis preneur.

J'ai essayé de faire tourner quelques utilitaires sur le XFormer et j'en ai trouvé deux. Le DISKFIX.COM du DOS 2.5 et le SHERLOCK 1050. Le Sherlock s'arrête lors du boot sur l'écran de présentation "ANTIC presents SHERLOCK", et il faut alors aller au bureau XFormer et cliquer sur CONTINUE. Les choses rentrent dans l'ordre. J'ai essayé de faire tourner TEXTPRO, ATARIWRITER+, SAM et Mini OFFICE II. C'est Mini Office II

qui donne les meilleurs résultats, la partie Spreed Sheet semble tourner, mais pas le reste. Autrement dit aucun de ces programmes ne fonctionne avec le XFormer. En ce qui concerne le BASIC, mon programme utilise quelques POKE et cela fonctionne. Par contre il ne faut pas utiliser l'instruction CLR, elle plante le XFormer. Mais mon seul but était de faire des transferts de fichiers ST vers XE, et j'ai pleinement réussi. Ca marche aussi dans l'autre sens, d'ailleurs le texte que vous lisez a été écrit sur 800 XL avec Textpro, puis modifié aux caractéristiques PC, toujours avec Textpro. Ensuite il a été transféré sur une disquette ST, qui peut être lue par un PC.

TEST DE JEUX
Par Daniel Carrodano

ADAX

Un jeu assez récent! Celui-ci, nomme ADAX, est une aventure d'arcade d'Avalon en Pologne et a été importé au Royaume Uni par Micro Discount, qui se spécialise dans l'offre de produits inédits. Bien que le programme contienne quelques indications en polonais cela n'est pas un problème car la plupart sont instinctives a comprendre et une traduction en anglais des instructions du jeu est fournie.

Il semblerait que des extraterrestres aient envahi la proche planète ADAX et les forces de défense spatiale ont élu leur meilleur élément - vous bien sûr - pour rendre compte de la situation. Il ne vous faudra pas longtemps pour découvrir que les immondes ont construit une base militaire et sont en train de préparer une attaque massive sur la Terre elle-même. Nul besoin de le dire, votre mission, si vous l'acceptez, sera de liquider le QG et voir tous ces salopards se débiner! Une bonne publicité pour VITTEL avec RAMBO en Star invitée: Il faut éliminer!

THE² CURSE

Si vous vous êtes lamenté sur la rareté des aventures graphiques dans le monde de l'Atari 8-bits vous serez heureux de faire connaissance avec la dernière découverte de Micro Discount. Klatwa de la société polonaise Avalon - ou LA COURSE pour ceux d'entre vous présentant des lacunes en polonais - est une aventure graphique "pointe et clique" présentant une généreuse variété d'objets et d'emplacements. Si, comme moi, vous n'êtes pas accoutumé aux aventures graphiques, vous pouvez être tenté de survoler les nombreux graphiques, mais vous perdrez alors beaucoup de l'intérêt présenté par ce jeu. Donc, le pays a été soumis par le maître des ombres, le plongeant dans le malheur et la misère. Par chance, vous avez pu repérer le château du maître, et vous avez maintenant l'opportunité de débarrasser votre pays de son terrifiant pouvoir.
L'écran est divisé en deux sections horizontales. La partie du haut montre votre personnage circulant dans les nombreuses pièces du château, tandis que dans la fenêtre du bas apparaissent les divers messages utiles. Les salles du château sont représentées dans une perspective 3D. En utilisant la manette pour déplacer un pointeur à l'écran, vous pouvez sélectionner chaque élément d'une scène pour l'examiner en détail. L'appui sur le bouton de feu fait apparaître un menu vous informant s'il y a quelque chose d'utile à la position indiquée. Si une action est jugée opportune, comme de regarder par une fenêtre, ou ouvrir une porte, votre personnage se déplacera pour l'accomplir. De nombreux objets sont nécessaires pour circuler à travers les scènes, mais vous ne pouvez en porter que cinq à la fois. Vous pourrez aussi recourir à des formules magiques pour contourner certains problèmes, mais celles-ci figurent dans votre livre de magie, alors n'encombrez pas votre inventaire.
Heureusement, ça ne semble pas être une de ces aventures dans lesquelles votre personnage expire à chaque faux pas. Dans certains cas vous pouvez perdre vos pouvoirs magiques et devrez alors chercher le moyen de les recouvrer; mais une option de sauvegarde de la partie en cours est fournie, ce qui pourra diminuer votre frustration.

PARTAGER UNE IMPRIMANTE ENTRE DEUX ORDINATEURS

Article de Daniel Carrodano

Si vous êtes assez chanceux pour posséder un autre ordinateur en plus de votre XL/XE, et que vous possédez une imprimante, vous en aurez vite assez de débrancher et de brancher un câble chaque fois que vous voulez changer de machine. La solution est simple et passe par un boiter de partage, avec deux ports en entrée, un pour chaque ordinateur, et une sortie dirigée vers l'imprimante. Mais vous aurez aussi remarqué que le Classic ne gère pas le saut de ligne de la même façon que le ST et le PC. Bien sur, les imprimantes possèdent une option permettant de s'arranger avec ça, et il suffit de changer la position d'un simple interrupteur. Mais pour peu que cet interrupteur soit dans la machine, il vous faudra chaque fois ouvrir celle-ci.

La première idée qui vient l'esprit c'est de brancher un autre interrupteur en parallèle sur le premier et de l'installer en façade de l'imprimante. Mais il faudra toujours penser mettre cet interrupteur sur la bonne position. Sinon, soit vous aurez deux sauts de lignes chaque fin de ligne, soit tout ce que l'imprimante devra imprimer elle le couchera sur la même ligne. Et vous oublierez souvent de régler l'interrupteur... Alors vous vous prenez à souhaiter que le réglage de l'interrupteur soit automatique. Eh bien n'en rêvez plus, la solution existe! Comme vous le savez, les prises Centronics possèdent 36 broches, et même les prises DB 25, présentes sur la plupart des configurations, ont 25 broches, alors que l'imprimante n'en utilise jamais autant. Il vous suffit de repérer deux fils inutilisés par l'imprimante et de les débrancher de sur la prise Centronics. Il y a des lignes indispensables: La ligne 1 STROBE, les lignes 2 à 9 pour le transfert des données, la ligne 10 ACKNOLEDGE, la ligne 11 BUSY ne doivent surtout pas être touchées. Sur ma machine, j'ai remarqué que les lignes allant de 19 à 30 étaient toutes reliées à la masse. J'ai donc débranché les broches 21 et 22, que j'ai relié à la broche 20. Puis j'ai soudé deux fils en parallèle sur l'interrupteur gérant le saut de ligne LF pour LINE FEED de mon imprimante, et je les ai soudés, de l'autre extrémité sur les deux broches libérées auparavant de la sortie Centronics de la machine. Il m'a fallu ensuite ouvrir le boiter de partage et vérifier que les lignes 21 et 22 n'étaient reliées à rien sur l'interface parallèle de mon XE. J'utilise une A850 et elle ne possède même pas ces deux sorties là, il n'y avait donc rien à craindre. J'ai repéré quelle était la broche reliée au XE et j'ai mis un pont de soudure entre les broches 21 et 22. Ainsi, à chaque fois que je manoeuvre le contacteur du boiter de répartition je positionne correctement le LF. Il va de soi que pour que cela marche, l'interrupteur d'origine de l'imprimante doit rester en position ouverte. C'est tout, mais croyez-le cela apporte un plus de confort appréciable. Le boiter de répartition, switcher box en anglais est un petit coffret existant en deux versions. Il en existe pour relier deux imprimantes à un ordinateur, mais le modèle qui nous occupe est celui qui permet de partager une imprimante entre deux machines.

Le prix de ce coffret peut varier de 800F, pour un modèle avec les contacts plaqués or, à 99F pour le modèle que j'ai acheté à INTERDISCOUNT. Au dos de ce boiter il y a typiquement trois prises, deux entrées (une pour chaque ordinateur), et une sortie allant à l'imprimante. En façade il y a un bouton qui manoeuvre le rotateur qui se trouve à l'intérieur. Un rotateur est un interrupteur/inverseur à multiples contacts tous basculés ensemble par la manoeuvre du bouton. Il ne vous reste plus qu'à coller une étiquette portant mention de chaque ordinateur en fonction de la position du bouton, A ou B.

Peut-on automatiser encore plus cette opération de partage? Pourquoi pas? On peut très bien utiliser la broche 35 sur laquelle une tension de 5 Volts est présente afin de distinguer quel est l'ordinateur présent. Il faudra bien sur gérer les conflits si les deux ordinateurs sont sous tension au même moment. Le plus dur sera de fabriquer un commutateur électrique ou électronique possédant le nombre de contacts voulus. Personnellement je me contente de poser le boiter de répartition près de l'écran, à portée de main, et cela suffit à mon bonheur.

LES JEUX D'ACTION

Sélection et mise en page par Rémi Gallopin

INCOMING!

Jeux écrit par Conrad Tatge pour **ANALOG**.

Vous devez protéger votre ville contre une attaque aérienne d'hélicoptères et de vaisseaux spatiaux.

Les hélicoptères larguent des parachutistes qui envahissent les toits de votre ville, les vaisseaux spatiaux bombardent votre ville. Mais attention, ce sont des bombes d'un type spécial, puisqu'elles font pousser les immeubles!

Votre seule défense est une pièce de DCA ayant une grande cadence de tir. Jusqu'a 16 missiles peuvent être visibles à l'écran. Quand vous ne pouvez plus tirer, le canon devient brillant. Pour diriger le tir, utilisez le manche du Joystick. Chaque tir retire 10 points de votre score.

Vous pouvez descendre les parachutistes en les abattants ou en dégommant leur parachute. Les hélicoptères et les vaisseaux spatiaux peuvent aussi être détruits, mais il faut tirer sans cesse dessus pour en faire exploser un. Les débris d'une explosion détruisent tout ce qui est autour.

Le jeu est terminé quand votre canon est détruit ou si plus de la moitié des immeubles sont occupés par des parachutistes. Appuyez sur START ou sur le bouton du joystick pour relancer le jeu.

Stratégie:

Un parachutiste sans parachute est comme mort, car il va chuter sur les immeubles et détruire d'éventuels autres parachutistes qui seraient en dessous.

Il y a deux type de vagues qui se succédant alternativement. Les vaisseaux qui lâchent des bombes vont en même temps détruire des parachutistes déjà arrivés sur les toits.

Mais ne laisser pas les immeubles trop pousser, car vous augmentez les chances des parachutistes de bien atterrir, et vous serez gêné dans vos tirs.

Elevator Repairman

Jeux écrit par Fred Caprilli pour **ANALOG**.

Vous êtes **Dan** le Réparateur d'ascenseurs, et vous allez avoir du pain sur la planche!

Vous avez été appelé d'urgence au *Polychromatic Hôtel* pour dépanner les ascenseurs qui ont été sabotés et évoluent anarchiquement.

Le problème: l'hôtel a été conçu par un architecte pas très compétent et pour gagner la salle de contrôle, il faut grimper sur le toit de l'immeuble, en prenant les escaliers de service qui sont à chaque extrémité d'étage.

La difficulté sera donc de traverser chaque étage en évitant les ascenseurs dans leur course folle, pour gagner le coté droit du haut.

Si un ascenseur vous touche, vous êtes renvoyé au début du même étage. Quand vous atteignez l'extrémité d'un étage vous montez automatiquement à l'étage supérieur.

Et quand vous atteignez le toit, vous repartez pour un autre niveau, avec des ascenseurs encore plus fous, et ainsi de suite.

Si vous atteignez le niveau 7, une surprise vous y attend.

Le haut de l'écran affiche un compteur qui démarre à 250 et mets 25 secondes pour atteindre zéro. Si vous atteignez l'extrémité de l'étage avant qu'il ne soit à zéro, vous marquerez un bonus supplémentaire.

Utilisez le joystick pour vous déplacer de droite à gauche, mais vous ne pouvez pas vous arrêter à un endroit donné, ce serait trop facile! On peut toutefois immobiliser Dan en manoeuvrant rapidement la manette de droite à gauche. Vous démarrez le jeu avec 9 vies, ce qui ne sera pas de trop, d'autant qu'on ne regagne jamais de nouvelles vies. Si vous regardez attentivement, vous verrez que le déplacement des ascenseurs est aléatoire et que la cadence évolue, ce qui évite un coté trop répétitif du jeu.

Bonne chance

ROTO

Jeux écrit par Mike Storz pour **ANALOG**.

Dans les cavernes d'Arcadia, des "cannisters" de fuel ont été stockés pour alimenter les boucliers qui protègent la cité. Comme Arcadia vient d'être attaquée par des aliens, vous avez été désigné pour aller rechercher ces cannisters et alimenter le bouclier.

Appuyez sur START ou sur le bouton du Joystick pour commencer le jeu. Vous verrez alors une portion de caverne et 4 blocks vert marqués "H". C'est par là que vous alimentez le bouclier en fuel. Des "cannisters" sont répartis dans les caves, ils ressemblent au fuel sauf qu'ils sont "glowin". Touchez-les pour les ramasser et quand vous en avez quelques uns, retournez à la réserve de fuel pour les y déposer. Cela accroit votre score et recharge votre bouclier.

Chaque "cannister" vaut 50 points, attendez d'en avoir ramassé 10 pour aller les déposer. Si vous en prenez plus de 10, vous prenez des risques...

Votre hélipack peut se déplacer dans toutes les directions. Quand vous arrivez au bord de l'écran, un scrolling vous montre le reste des cavernes. Quand ça ne "scrolle" plus c'est que vous êtes à l'extrémité des cavernes. Ne touchez pas les murs sinon vous perdez un hélipack. Faites attention à votre réserve de fuel aussi...

Vous êtes aussi équipé d'un laser pour percer les murs des cavernes. Mais n'en abusez pas trop, car vous perdez des points à chaque tir. Si vous tirez sur un "cannister", il va exploser! Et si vous êtes trop près, vous serez détruit.

Pendant le jeu, appuyez sur une touche pour l'interrompre momentanément. Rappuyez sur une touche pour reprendre le cours du jeu. Appuyez sur START pour reprendre le jeu au début.

Les cavernes sont générées aléatoirement à chaque partie. Rappelez-vous les parties que vous avez déminées après être retourné déposer des cannisters.

Le jeu peut se terminer de plusieurs manières:

Si vous avez perdu tout vos hélipack.

Le bouclier est à zéro
Un cannister est touché
Vous transportez plus de 10 cannisters.
Et si vous avez récupéré tous les cannisters...
Arcadia est sauvée.

SUPER PONG

Jeux écrit par Gary Domrow pour **ANALOG**.

Les adaptations du jeu de Ping-pong étant pratiquement absentes sur le XL, G. Domrow a donc décidé d'y remédier en adaptant ce classique du jeu vidéo sur le 8 bit.

Super Pong nécessite une commande à molette (ou paddle) pour fonctionner.

Après le chargement, appuyez sur le bouton du paddle pour commencer. Tournez le bouton du paddle pour choisir le type de jeu et appuyez à nouveau sur le bouton.

Un point est marqué chaque fois qu'un joueur manque la balle. Le premier joueur à marquer 16 points est le gagnant. Pendant le jeu, appuyez sur une touche pour l'interrompre momentanément. Appuyez sur START pour reprendre le cours du jeu.

Quatre jeux différents sont proposés, Standard pong, Hockey Pong, Wall Pong et Practice. Les 3 premiers sont pour deux joueurs uniquement, Practice est une version un joueur de Wall Pong pour l'entraînement.

Suite de "Variable Name Table Editor"

La ligne 1210 lit l'ancien en-tête et la VNT de l'ancien fichier et les écarte. La ligne 1230 est lit un octet sur l'ancien et l'écrit sur le nouveau tant que la fin de l'ancien n'est pas atteinte. Les autres parties du programme sont auto-explicatives.

VNT Editor vous fournira un nouveau moyen de voir et manipuler la VNT de vos programmes BASIC. Ce contrôle et cette flexibilité ajoutés seront très appréciés dans de nombreux programmes et seront très utiles dans les plus grands de ceux-ci utilisant un nombre important de variables.

NDLR: *VNTEDT.BAS, bien que similaire à HELPER.BAS (décris dans ce même numéro) lui est en fait complémentaire.*

AJOUT DE TOUCHES DE FONCTION AU BASIC

par Stephen Propkopchuk
ANALOG COMPUTING No 25
traduit par Daniel CARRODANO

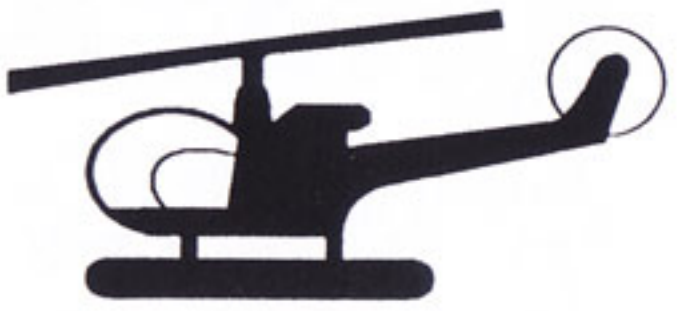
Un jour, en tapant un long programme, je réalisais combien souvent on tapait encore et encore les mêmes mots. Si ces mots pouvaient être écrits en pressant une simple touche, je songeais que beaucoup de temps pourrait être économisé.

Le programme qui en résulta définit CTRL 4-0 et SHIFT+CTRL 4-0 comme touches de fonction. Il se déroule à travers le vecteur d'interruption de clavier POKLEY VKEYBD rarement utilisé (adresses mémoires 520-521 ($208-209 hex)). Chaque fois qu'une touche est pressée, l'ordinateur arrête ce qu'il est en train de faire et saute à la routine de balayage du clavier normalement située en 65470 ($FFBE hex). Cette interruption a été déroutée pour notre programme, **Function Keys**, lequel contrôle si une de ces touches a été pressée.

COMMENT L'UTILISER

Deux programmes existent, un en BASIC et un en assembleur. Seul celui en BASIC figure sur la disquette (FUNCTION.BAS), une copie du code source assembleur pouvant être fournie à quiconque en fera la demande. Après avoir lancé le programme BASIC Function Keys, votre ordinateur vous donnera de brèves instructions et vous montrera le paramétrage par défaut (voir Table 1), pour la touche de fonction #1 (LOAD), vous demandant votre choix. Si vous voulez que cette touche garde le paramétrage par défaut, pressez simplement RETURN.

Si vous voulez assigner votre propre mot à cette touche, tapez le mot puis RETURN. Gardez à l'esprit que la longueur du mot ne doit pas excéder sept lettres. L'ordinateur vous affichera des indications similaires pour chaque touche pouvant être programmée



Commande	Fonction
CTRL_4	LOAD"
CTRL_5	SAVE"
CTRL_6	LIST
CTRL_7	RUN
CTRL_8	POKE
CTRL_9	PEEK(
CTRL_0	DATA
SHIFT+CTRL_4	PLOT
SHIFT+CTRL_5	DR.(DRAWTO)
SHIFT+CTRL_6	LOC.(LOCATE)
SHIFT+CTRL_7	POS.(POSITION)
SHIFT+CTRL_8	COL.(COLOR)
SHIFT+CTRL_9	SE.(SETCOLOR)
SHIFT+CTRL_0	GR.(GRAPHICS)

Quand le BASIC affiche READY, vos touches de fonctions sont toutes définies. Essayez en pressant CTRL_4 à CTRL_0 ou SHIFT+CTRL_4 à SHIFT+CTRL_0 et regardez les mots que vous avez défini apparaître à l'écran.

Gardez présent à l'esprit qu'une remise à zéro du système déconnectera les touches de fonction. Pour les remettre en action, tapez:

PRINT USR(1536)

De même, pressez RESET avant de relancer le programme BASIC sinon il affichera FUNCTIONS ALREADY IMPLEMENTED ("fonctions déjà implémentées").

Ce programme n'est qu'un petit exemple de ce qui peut être fait par l'utilisation les interruptions clavier. Le clavier pourrait être reconfiguré des centaines de façons, y compris bouger les touches de déplacement du curseur.

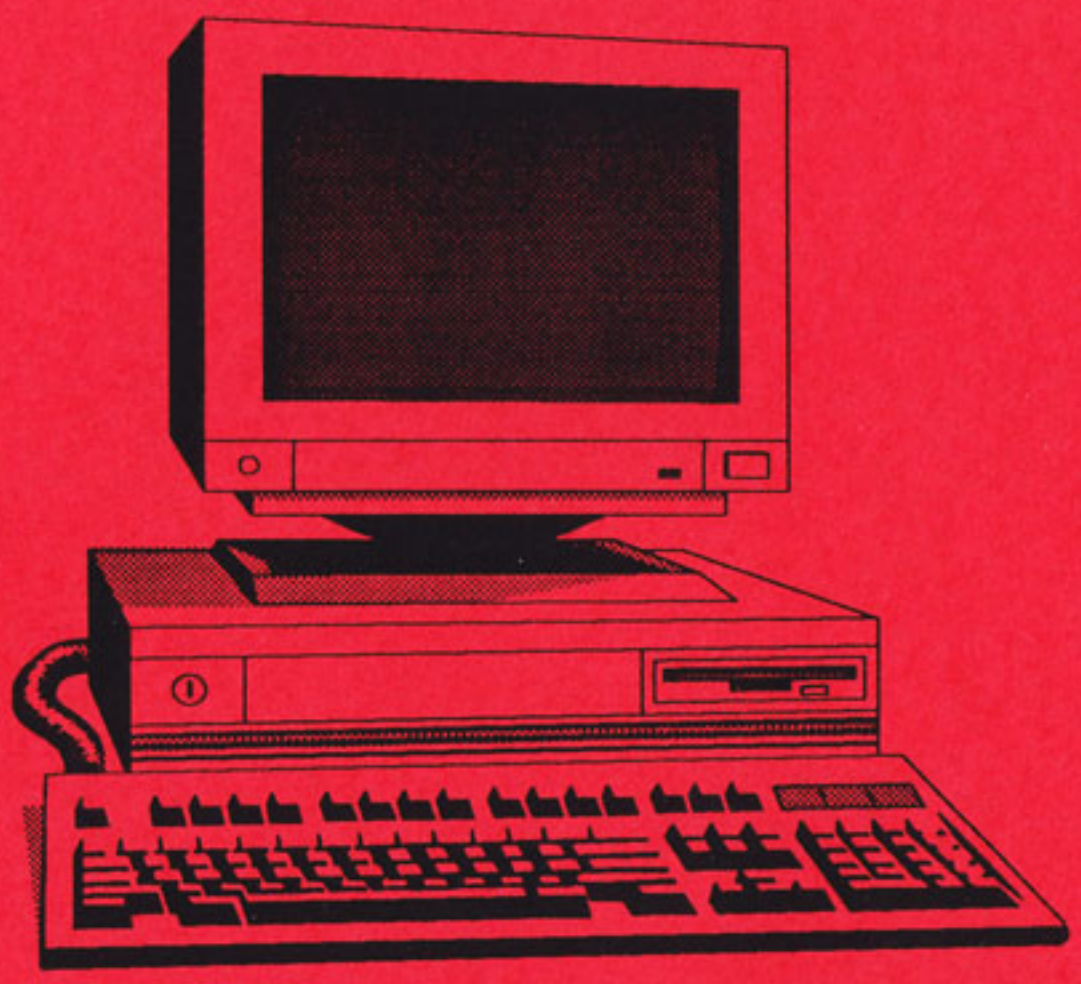
Turbo Basic version SP 3.2

Dans le fichier ARChive BINARY.ARC, vous trouverez une version du Turbo-Basic adaptée à Sparta-DOS 3.2 (lire le fichier texte ITB32Q.TXT pour plus d'informations).

Pour utiliser ce turbo-Basic, bootez d'abord votre XL/XE avec Sparta-DOS 3.2 ou X, puis tapez ITB32Q sur la ligne de commande.

Attention, il s'agit ici d'une version 48 Ko du Turbo, et qui vous laisse 23 Ko de Free Memory pour vos programmes.

Publications du Club Cenacle:



Cenacle-News No 4: Avril 1988 *56 pages, 1 disk* 35.00 Frs
Dossier sur l'imprimante graphique 1020
- pleins de programmes pour votre 1020 dont TURTLE 1020 pour programmer votre plotter comme en Logo.
- Formation à SYNFILE+

Cenacle-News No 5: Décembre 1988 *28 pages, 1 disk* 25.00 Frs
Utilitaires Basic: MicroDos XL, Instant Dos 2.5, etc.
Solution de FREE

Cenacle-News No 6: Juillet 1989 *34 pages, 1 disk* 30.00 Frs
Basic Package de Pierre Faller
Utilisation de l'assembleur MAC 65

Cenacle-News No 7/8: Avril 1990 *64 pages, 2 disks* 45.00 Frs
- Dossier imprimantes matricielles Epson & 1029
  GHANDLER pour faire des recopies d'écran
  K7JACKET pour vos jaquettes de K7
  Designer Labels, Edit Magic, Star Rider
- La mémoire des Atari XL/XE

Cenacle-News No 9: Décembre 1990 *44 pages, 1 disk* 35.00 Frs
Editeur de jeux de caractères avec visualiseur et exemples
Utilisation de MAC 65 part 2
Utilitaires de programmation: LISTIF, Number Editor, etc.

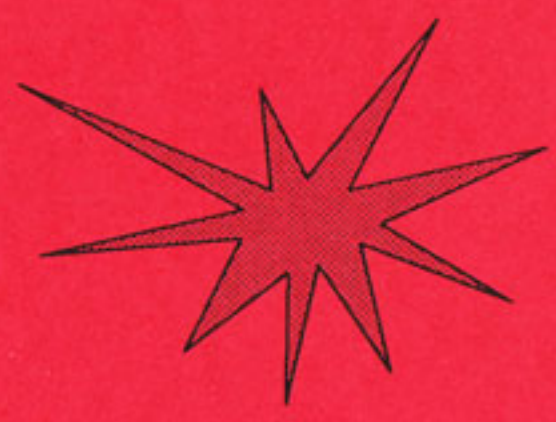
Cenacle-News No 10: Novembre 1991 *80 pages, 2 disks* 50.00 Frs
Dossier lecteur de diskettes et DOS
- Comprendre comment ça marche, qu'est-ce qu'un DOS
- Diskmend: un éditeur de secteur,
- BBKCP: Ajoutez un command Processeur au DOS 2.5

Kit bureautique comprenant tableur, Base de données, traitement de texte et programme de dessin
Utilisation de MAC 65 part 3

Cenacle-News No 11: Juillet 1992 *64 pages, 1 disk* 40.00 Frs
Gestion du graphisme sur imprimante 1029
Utilisation de Sparta-DOS et Super DOS 4

Spécial Quick No 1: Spécial jeux de caractères (16 pages) 15.00 Frs
Spécial Quick No 2: Spécial modes graphiques (32 pages) 25.00 Frs

GHARLE



JOYEUX NOËL
A TOUS !!!