

# **tremplin micro**

**C TRÈS SIMPLE :**  
apprenez le langage C  
sur le GS !

**La souris  
et l'assembleur**

**Quelle heure est-il  
dans le Monde ?**

**DOS 3.3 pas mort**

**Les carrés magiques  
en Basic**

**Amortissement**

**Fenêtres HGR**



M 1631 - 14 - 33,00 F



3791631033006 00140

N° 14 - Bimestriel - Troisième année  
5 Mai - 4 Juillet 1987

254 FB - 11 FS - **33 F**

# tremplin micro 14

# SOMMAIRE

Avec la collaboration de :

Argos, Claude Aubry, Nicole Bréaud-Pouliquen, Robert Cazenave, Marcel Cottini, Jacques Fourneau, François Gallet, Yvan Hué, Roland Jost, Yvan Kœnig, Nestor, Jean Perrot et Clément Renard.

Apple et ProDOS (noms et logos) sont des marques déposées d'Apple Computer, Inc.

## BIMESTRIEL

Le numéro : 33 F  
Abonnement d'un an : 190 F  
(6 numéros)

Tous nos prix sont indiqués TTC.

## EDITIONS JIBENA

Direction-Rédaction :

Editions JIBENA

Guy-HACHETTE

La Petite Motte — Senillé  
86100 CHÂTELLERAULT.

Téléphone :

49-93-66-66

PUBLICITÉ :

Raymond JULLIEN  
(1) 45.75.41.81

Commission paritaire :

Les revues qui choisissent d'être réellement au service du Lecteur, en ne l'obligeant pas à glaner, dans plusieurs magazines, les renseignements concernant sa machine, ne bénéficient pas du numéro de Commission Paritaire, et pas davantage des tarifs postaux réduits.

TREMPIN MICRO — Bimestriel — C'est une publication des Editions JIBENA, 4, rue de la Cour-des-Neuves, 75020 PARIS — S.A. au capital de 3600000 F — Imprimé par CITÉ-PRESS/PARIS — Dépôt légal à la date de parution — Inscription à la Commission Paritaire des Publications et Agences de Presse : en cours — Directeur de la Publication : Guy-Clément COGNÉ — Diffusion N.M.P.P.

La Disquette TREMPIN MICRO contient tous les programmes du numéro, ainsi que les sources trop longs pour être publiés dans les colonnes de la revue.

LIBRES PROPOS .....	2
HEURES (Quelle heure est-il dans le Monde ?) .....	3
LES TRANSFERTS DE VARIABLES (Basic vers LM) .....	9
VIDEC (Effacement progressif de l'écran) .....	12

<b>C TRÈS SIMPLE</b> (Apprenez le langage C sur le GS) .....	13
<b>LE BON NOMBRE</b> (Exercice élémentaire en C) .....	17

GAGNEZ UN APPLE AVEC VERSION SOFT... ET APPLE .....	18
LEÇONS DE LOGIQUE AVEC NOS AMIS BELGES .....	19
DOS 3.3 PAS MORT : VOICI UN PATCH .....	21
INKEY (Pomme ouverte et pomme fermée) .....	27
LES CARRÉS MAGIQUES (En Basic, mais complet) .....	30
FOND D'ÉCRAN (65C02) .....	34

<b>SPÉCIAL</b> NON-ACCÈS AU TABLEAU DE BORD .....	35
<b>GS</b> TIC-TAC (Le cœur de votre GS bat) .....	37

CRÉATION DE CARACTÈRES (Gribouille vous attend) .....	39
QUELQUES ADRESSES INTÉRESSANTES DES APPLE .....	40

<b>SPÉCIAL</b> UN BUG CORRIGÉ DANS L'APPLESOFT DU GS .....	41
<b>GS</b> TESTEZ LE MODE GR DU GS .....	42

BON À SAVOIR (Tridos — Toujours le langage C) .....	43
CORRIGEZ MERLIN POUR L'UTILISER SUR LE GS .....	44
LE PRIX MANNESMANN TALLY 1987 .....	46
TRANSBIN (Déplacement de blocs sur le GS) .....	47
ALEALIGNÉS (Amusette en mode HGR) .....	49
AMORTISSEMENT (Utilitaire en Basic) .....	52

<b>LA SOURIS ET L'ASSEMBLEUR</b> (Deuxième partie) .....	55
--	----

SONSIR, MAIS EN COMMANDES EXTERNES PRODOS .....	60
PRO.FP (Source de la nouvelle version) .....	64
FENETRES HGR .....	67
LES LIVRES .....	11, 16, 20, 43, 76
YVAN KOENIG RÉPOND À NOS LECTEURS .....	72
BULLETIN DE COMMANDE .....	75

# LIBRES PROPOS

## Avez-vous troqué votre APPLE // contre un GS ?

J'espère que, mieux informé que *Tremplin Micro*, vous avez pu bénéficier de la dernière offre d'Apple Computer France et abandonné votre vieil Apple // (+ ou e) contre un GS tout neuf. Rassurez-vous, rien n'est perdu : vous avez jusqu'au 16 juin (c'est un mardi... allez savoir pourquoi on a choisi cette date plutôt qu'une autre) pour vous offrir un 65C816. Il vous en coûtera 6523 F TTC si vous choisissez la configuration TGS (unité centrale, clavier, 512 K et souris).

Vous aurez au préalable remis votre ancienne unité centrale à votre revendeur qui, bien que cette condition figure dans l'offre de reprise, n'en aura probablement pas vérifié le bon fonctionnement. Vous aurez aussi conservé vos cartes et périphériques, mais en sachant que la compatibilité avec le GS ne sera pas forcément exempte de problèmes.

Votre revendeur possède une liste des produits réellement compatibles. Ils sont nombreux et la plupart de vos anciennes cartes devraient vous autoriser à utiliser vos antiques périphériques sur le GS (l'imprimante DMP Apple fonctionne normalement avec sa carte parallèle dans le slot 1).

### OUI OU NON AU GS ?

Si vous disposez de la somme indiquée plus haut (et si vous acceptez de "travailler" avec votre moniteur monochrome), n'hésitez pas : changez votre Apple, surtout s'il s'agit d'un Europlus. Vos programmes personnels y gagneront en rapidité d'exécution et vous aurez le plaisir de disposer d'un processeur actuel, puissant, dont les possibilités sont impressionnantes.

Quant à la machine, il n'est pas nécessaire de revenir sur ce qui en a déjà été dit ici : c'est une excellente bécane, digne de la Pomme qu'elle affiche. Peut-être est-il dommage que les bouleversements survenus chez Apple aient quelque peu retardé sa mise au point et stoppé le dynamisme des développeurs, très occupés par ailleurs. Il convient de ne jamais oublier que, dans le pire des cas, un GS se comporte encore mieux (beaucoup mieux) que ses illustres prédécesseurs. Pour obtenir, sur un Ap-

ple //, les résultats notés sur un GS, il faudrait actuellement investir, en cartes additionnelles (et je ne parle pas de l'accélérateur !), une somme au moins équivalente à celle qui est demandée pour échanger l'ancien matériel contre le nouveau.

### La tentation du compatible

D'accord, il est vrai que pour le même prix on peut s'offrir un compatible PC bas de gamme, mais en sachant que, pas plus tard que demain, ce standard sera probablement renié par IBM.

Reste que l'Applesoft n'arrive pas à la cheville du Basic moyen disponible sur compatible. On sait qu'un Basic permettant d'accéder facilement aux outils est en gestation pour le GS, mais on (ON, c'est Apple) ne le dit pas, ce qui constitue assurément une grave erreur : le public a besoin d'informations sur les matériels qu'il achète. C'est son droit. Et c'est un devoir, pour les fabricants, d'accéder à ce légitime désir. Ceci étant, *Tremplin Micro* vous conseille d'apprendre à programmer en C... et vous y aidera. Mais ça ferait toujours plaisir d'apprendre officiellement que, dans *n* mois, un Basic puissant sera à la disposition des heureux (le sont-ils ?) possesseurs d'un GS.

### Et les prix ?

Là, c'est une autre histoire. On peut même affirmer que c'est le défaut de la cuirasse. C'est là que le bât blesse. Il blesse surtout les ânes (j'en fais bien sûr partie) qui, périodiquement, se font avoir en essayant les plâtres. Jusqu'à maintenant, côté GS, rien à dire. Du moins quand on a pu bénéficier de la remise de Noël du club Apple... parce que chez les autres, je veux dire les gogos qui ont payé leur GS au prix fort, l'arête s'incruste aux amygdales ou à la luette, non sans provoquer les réactions que l'on devine... et que l'on comprend.

Chacun est convaincu que, après la campagne de reprise des Apple défraîchis, le prix du GS baissera... du montant de la reprise. Ça se passera en juillet ou en octobre. Ça commencera bien sûr par une offre destinée aux seuls enseignants (oh ! la belle injustice !)... puis, la baisse du dollar aidant, à

moins qu'il ne s'agisse de l'apparition d'un GSplus (déjà !), on bradera les machines en stock. Ne jetons pas la pierre aux fabricants : c'est devenu une habitude. C'est la loi du marché. Je crains pourtant que, petit à petit, les consommateurs avertis n'en arrivent à boudier systématiquement les nouveaux produits.

### L'Apple //c

Ayant bénéficié — au contraire de l'Apple IIe — d'un important effort promotionnel, l'Apple //c est une machine très répandue. On l'affectionnait beaucoup chez Apple : c'est un produit fermé (mais c'était sans compter sur l'imagination des bidouilleurs de génie), bien conçu pour transformer l'utilisateur en mouseman impénitent, condamné à passer d'un logiciel à fenêtres à un autre logiciel à fenêtres... bref fin prêt pour plonger dans l'univers chéri d'Apple : le macworld. Foin des programmeurs amateurs ! Vive LE LOGICIEL UNIVERSEL, pensant à votre place. Apple apprend l'Homme ? non : l'Homme devient l'esclave de la machine. Horrible destin ! Mais on en revient. Le Mac s'ouvre sur l'extérieur. Le GS n'en finit plus de compter ses slots et IBM — qui se moque complètement des amateurs — n'a pas négligé ce détail important : un ordinateur dit personnel ne doit pas être une machine fermée. Mais revenons au //c. La dernière version est assurément intéressante, mais pour combien de temps ? Qui ignore encore que les jours du 65C02 sont comptés ? Ou l'Apple //c nous présentera sa version TURBO ou il sera transformé, ici et là, en Minitel intelligent.

### Tremplin Micro

Revue de programmes, *Tremplin Micro* ne modifiera pas sa ligne de conduite. On connaît, dans le domaine de la chanson, par exemple, quelques grandes vedettes. Nous nous contentons en général de les écouter, et parfois de siffloter les refrains de leurs principaux succès. Mais il arrive aussi que certains d'entre nous se mettent à chanter, à la fin d'un repas... ou comme cela, pour le plaisir, en se rasant ou en pédalant. Pour l'informatique, c'est la même chose, les programmeurs amateurs existent et j'ai le plaisir d'être très souvent en rapport avec eux. Certains ne désirent que s'amuser à aligner des instructions, heureux ensuite de contempler le déroulement de leur petite routine. D'autres vont plus loin et il leur arrive même de passer dans l'autre camp, celui des gens de métier. Les uns et les autres ont en commun le désir d'approfondir, de faire travailler leurs neurones... avec ou sans souris, avec ou sans la convivialité à laquelle je préférerais, aujourd'hui, non seulement chez Apple, mais chez tous les fabricants, la COMMUNICATION.

Guy-HACHETTE.



# HEURES



Voulez-vous jouer avec les heures et les fuseaux horaires ? Si la longueur du programme (cent pour cent en Basic Applesoft) d'Yvan HUE ne vous décourage pas, mettez-vous d'abord au travail, puis étonnez vos parents et amis par votre "Science" !

Vous constaterez qu'il y a en fait quatre programmes tournant autour de la question. Sans doute aurait-on pu les séparer en conservant leur partie commune. Libre à vous de modifier, comme vous l'entendrez, le long programme d'Yvan HUE.

Notez que, dans l'état actuel d'HEURES, une carte 80 colonnes est indispensable. Nous vous conseillons par ailleurs ProDOS, beaucoup plus rapide que le DOS 3.3 lors du chargement.

```

110 DATA "ABIDJAN",0,"ACORES",-2
    ,"ADELAIDE",9,"ALGER",0,"ANG
OLA",1,"BUENOS AIRES",-4,"CA
LCUTTA",6,"DELHI",5,"DENVER"
,-7,"ILES MARQUISES",-10,"LE
CAIRE",2,"LENINGRAD",2,"LIM
A",-6,"LONDRES",0,"MEXICO",-
7
115 DATA "MOSCOU",2,"NEW YORK",-
5,"NOME",-11,"NOUMEA",11,"NO
UVELLE ORLEANS",-6,"NOVOSIBI
RSK",6,"PEKIN",8,"PERTH",8,"
POINTE EST DE SIBERIE",12,"P
RAGUE",1,"REYKJAVIK",-1,"RIO
DE JANEIRO",-3
120 DATA "SAIGON (HO-CHI-MINH-VI
LLE)",7,"SAN FRANCISCO",-8,"
STOKHOLM",1,"SVERDLOVSK",4,"
SYDNEY",10,"TAHITI",-10,"TAN
ANARIVE",3,"TEHERAN",3,"TERR
E-NEUVE",-4,"TOKIO",9,"VLADI
VOSTOK",9,"VOLGOGRAD",3,"WEL
LINGTON",12
125 PRINT CHR$(4);"PRÉ3": PRIN
T
130 CLEAR : HOME : VTAB 6: POKE
36,36: PRINT "MENU"
135 PRINT : PRINT : HTAB 10: PRI
NT "1 = Horaire d'atterrissa
ge": PRINT : HTAB 10: PRINT

```

2333

0967

2551

7091

DAFE

"2 = Calcul de l'heure en un point dont vous indiquez la longitude"

A340

140 PRINT : HTAB 10: PRINT "3 = L'heure d'un lieu du tableau inclus dans le logiciel"

CF60

145 PRINT : HTAB 10: PRINT "4 = Longueur d'un parallèle et l'argeur d'un fuseau en Kilomètres"

7369

150 PRINT : HTAB 10: PRINT "5 = Terminer"

FC3B

155 B\$ = "5": GOSUB 975

273E

160 D\$ = "":PI = 3.14159265

CA21

170 C = VAL (C\$): ON C GOTO 175,1020,1335,1240,555

ABCE

175 HOME

2F97

180 HTAB 29: PRINT "HORAIRE D'ATERRISSAGE"

615C

185 VTAB 4: PRINT "Soit un avion qui décolle d'un aérodrome situé sur le méridien de Greenwich à": PRINT : PRINT "une latitude que vous choisirez."

8F45

195 VTAB 9: PRINT "Nous allons calculer à quelle heure il se posera sur le terrain d'arrivée selon"

F354

(suite page 4)

# HEURES

(suite)

205 PRINT : PRINT "la direction de celui-ci."	6362	345 PRINT "Direction " "Atte	50!
210 VTAB 14: PRINT "Les calculs seront basés sur les heures solaires en raison de la diversité des"	8A8B	rrissage " "Date " "Observation"	50!
220 PRINT : PRINT "heures légale s (ainsi en France l'heure officielle avance en été de 2 heures et"	B9D8	350 M = 0: IF D < = 0 THEN GOS	51!
225 PRINT : PRINT "en hiver de 1 sur l'heure réelle)."	4EA9	UB 1015: GOTO 305	51!
230 VTAB 21: PRINT "Les calculs sont arrondis à la minute."	575B	355 IF D > = 40000 THEN M = D /	
235 VTAB 24: GOSUB 685	6895	40000	1268
240 GOSUB 570	ED4C	360 T = D / V	1C89
245 IF P = 90 THEN GOSUB 910: G	BDBF	365 Q = 24 / (40000 * COS (LT))	D43E
OTO 240		* D	
250 LT = (P + PM / 60) * PI / 18	0A6A	370 A = H + T: GOSUB 775:G = LT	2612
0:L1 = P:L2 = PM	0F09	* 40000 / PI / 2	52!
255 IF W = 1 THEN GOTO 320	0F09	375 IF D - M * 40000 > 10000 - G	53!
260 HOME : VTAB 10:Y = 24	9F9D	AND D - M * 40000 < 30000 -	53!
265 PRINT "Quelle est l'heure de	ED1C	G THEN GOSUB 890	54!
départ ? "	1952	380 VTAB 10	54!
270 GOSUB 990	0A6A	385 DIR\$ = "le nord": GOSUB 820	
275 IF W = 1 THEN GOTO 320	0F09	390 A = H + T: GOSUB 775	
280 HOME : VTAB 12:Z = 46	ACA4	395 IF D - M * 40000 > 10000 + G	55!
285 PRINT "Quelle est, en Km, la		AND D - M * 40000 < 30000 +	55!
vitesse de l'appareil ? ":	630F	G THEN GOSUB 890	56!
GOSUB 700	FFBF	400 VTAB 13	
290 IF X = 0 THEN GOSUB 1015: G	A17E	405 DIR\$ = "le sud": GOSUB 820	57!
OTO 280	0F09	410 M = 0:F = 0	57!
295 V = X	AE9F	415 QQ = Q: IF Q > = 24 THEN QQ	58!
300 IF W = 1 THEN GOTO 320	E7F2	= Q - INT (Q / 24) * 24	58!
305 HOME : VTAB 12:Z = 50	FB6C	420 IF QQ > = 12 THEN F = F - 1	
310 PRINT "Quelle est la distanc	F010	: GOSUB 545	
e kilométrique à parcourir ?	F7BB	425 A = H + QQ + T: IF Q > = 24	59!
": GOSUB 700	FCC5	THEN M = Q / 24	
315 D = X	221F	430 VTAB 16	
320 HOME : PRINT "Données : "Lati	40DA	435 GOSUB 775:DIR\$ = "l'est": GO	59!
tude = ";L1;" " ";L2:"/": HTA		SUB 820	
B 11: PRINT "Heure de départ		440 IF QQ > = 12 THEN F = F + 1	60!
= ";H1;" H " ";H2;" m"		: GOSUB 545	60!
325 HTAB 11: PRINT "Vitesse moye		445 A = H - QQ + T	61!
nne = ";V: HTAB 11: PRINT "D		450 IF A - INT (A) > 0.99 THEN	
istance à parcourir = ";D;"		A = INT (A) + 1	61!
Km"		455 IF A < 0 THEN F = F - 1	62!
330 VTAB 6: POKE 36,23		460 IF A < 0 THEN A = A + 24: GO	62!
335 PRINT "HEURE D'ARRIVEE SELON		TO 460	63!
LA DIRECTION"		465 VTAB 19	
340 VTAB 8		470 DIR\$ = "l'ouest": GOSUB 775:	63!
		GOSUB 820	64!
		475 Z = INT ( INT ( INT ((T -	64!
		INT (T)) * 60) * 100) / 100)	65!
		480 VTAB 22: POKE 36,26: PRINT "	65!
		Durée du vol = "; INT (T);"	66!
		H " "; INT ( INT ( INT ((T -	
		INT (T)) * 60) * 100) / 100)	
		;" m";	
		485 GOSUB 685	
		490 W = 2	
		495 B\$ = "4": IF W = 2 THEN B\$ =	

*5"	DB61	23: POKE 36,1: PRINT "Il ne	
500 GOSUB 945	3552	peut y en avoir que ";Y;" !"	
505 C = VAL (C\$): ON C GOTO 510		: GOSUB 685	DE8B
,560,130,555,1510	A667	670 IF X < 0 THEN X = X - X * 2	7D67
510 HOME : VTAB 10: HTAB 8	ECDC	675 RETURN	63B1
515 PRINT "Voulez-vous ? 1 = Lat		680 PRINT : POKE 36,27: PRINT "	
itude": PRINT : HTAB 22: PRI		": POKE 36,37: PRINT ""	BAC9
NT "2 = Heure de départ"	2E2D	685 VTAB 24: POKE 36,1: PRINT "P	
520 PRINT : HTAB 22: PRINT "3 =		our continuer, frapper une t	
Vitesse": PRINT : HTAB 22: P		ouche quelconque autre que r	
RINT "4 = Distance à parcour	E48C	etour chariot ";: GET C\$: PR	
ir"	2F55	INT D\$;: IF ASC (C\$) = 13 T	
525 GOSUB 975	E058	HEN GOTO 685	3C77
530 W = 1	F5B0	690 VTAB 22: PRINT CHR\$ (11);	C3CF
535 C = VAL (C\$)	F486	695 RETURN	63B1
540 ON C GOTO 240,260,280,305		700 REM SAISIE DE VALEURS	
545 OBS\$ = "Ligne de changement	FC72	705 X\$ = ""	3A90
de ":AJ\$ = "date franchie"	63B1	710 VTAB 22: POKE 36,2: PRINT S	
550 RETURN		PC( 10): POKE 36,2	7E33
555 HOME : VTAB 12: HTAB 36: PRI	1C4B	715 FOR I = 1 TO 20	6EEE
NT "Terminé": END		720 GET C\$: PRINT D\$;: IF ASC (	
560 CLEAR :Z = 12:PI = 3.1415926	5E6F	C\$) = 21 THEN GOTO 720	0F7C
: GOTO 240		725 IF ASC (C\$) = 8 THEN GOTO	
570 REM SAISIE DES COORDONNEES	01BF	705	305E
575 Z = 12	C9A0	730 IF C\$ = "," THEN C\$ = "."	2EC1
580 HOME : VTAB 10:Y = 90		735 IF ASC (C\$) = 11 THEN GOTO	
585 PRINT "Indiquer la latitude	E2CE	755	8E8D
choisie (par exemple Paris e		740 IF ASC (C\$) = 13 AND X\$ = "	
st à 48° 50')	169B	" THEN GOTO 710	A7E3
590 VTAB 12: HTAB 20: PRINT "Deg		745 IF ASC (C\$) = 13 THEN GOTO	
rés = ";: POKE 36,40: PRINT		770	D28C
"Minutes =		750 PRINT C\$;: IF C\$ = "." THEN	
595 VTAB 12: POKE 36,28: PRINT "	1AFC	GOTO 760	C0F8
""";: POKE 36,28	6494	755 IF C\$ < "0" OR C\$ > "9" THEN	
600 Z = 28	FF47	VTAB 23: POKE 36,1: PRINT "E	
605 GOSUB 700		rreur de frappe": GOSUB 685:	
610 GOSUB 655: IF X = 1000 THEN	5432	GOTO 700	3CFB
GOTO 595	3578	760 X\$ = X\$ + C\$	36F7
615 P = X	0A30	765 NEXT I	9DCB
620 IF P = Y THEN GOTO 650	1860	770 X = VAL (X\$): RETURN	30C5
625 Y = 59:Z = 50		775 REM DETERMINATION DE LA DATE	
630 VTAB 12: POKE 36,50: PRINT "	63F2	780 DTE\$ = ""	FD15
""";: POKE 36,50	FF47	785 IF A - INT (A) * 60 = 60 TH	
635 GOSUB 700		EN A = INT (A + 1)	19B5
640 GOSUB 655: IF X = 1000 THEN	5428	790 IF A > 24 THEN A = A - 24:F	
GOTO 630	F0C5	= F + 1: GOTO 790	1C7C
645 PM = X	63B1	795 DTE\$ = "ce jour": IF F > 1 T	
650 RETURN		HEN DTE\$ = STR\$ (F) + " jou	
655 REM VERIFICATION DE VALEURS		rs après"	A3CC
660 IF X < > INT (X) THEN X =		800 IF F = 1 THEN DTE\$ = "le len	
1000: VTAB 23: POKE 36,1: PR		demain"	1B0B
INT "Donner un nombre entier		805 IF F = - 1 THEN DTE\$ = "la	
, s'il vous plait !": GOSUB		veille"	6704
685: RETURN	F36E	810 F = 0	3046
665 IF X > Y THEN X = 1000: VTAB			

(suite page 6)

# HEURES

(suite)

815 RETURN	63B1	PRINT : HTAB 4: PRINT "heure	105
820 REM AFFICHAGE		s puisque tous les méridiens	
825 PRINT DIR\$;: POKE 36,17	DFE8	y convergent."	EA8D
830 IF INT (A) < 10 THEN PRINT	2661	935 PRINT : PRINT : POKE 36,23:	106
" ";		PRINT "L'exercice n'a donc p	106
835 PRINT INT (A + 0.0001);" H	5868	as de sens.": GOSUB 685	F869
" ";		940 RETURN	63B1
840 A2 = INT ((A + 0.0001 - IN	D9CE	945 HOME : VTAB 10: HTAB 8	ECDC
T (A + 0.0001)) * 60): IF A2	CD54	950 PRINT "Voulez-vous : 1 = Mod	107
< 10 THEN PRINT " ";	EFE6	ifier une des données sans c	107
845 PRINT A2;" m";	360D	hanger les autres	3B7D
850 POKE 36,35: PRINT DTE\$;		955 PRINT : HTAB 22: PRINT "2 =	108
855 IF M = 0 THEN GOTO 875		Recommencer sur d'autres don	66E0
860 M = INT ((M + 0.00001) * 10		nées"	108
0) / 100:TR\$ = "tours": IF M	B685	960 PRINT : HTAB 22: PRINT "3 =	18DA
< 2 THEN TR\$ = "tour"		Revenir au menu	443D
865 OBS\$ = " du parallèle": IF D		965 PRINT : HTAB 22: PRINT "4 =	
IR\$ = "le nord" OR DIR\$ = "l	B685	Terminer"	443D
e sud" THEN OBS\$ = " de terr		970 IF W = 2 THEN PRINT : HTAB	109
e"	154E	22: PRINT "5 = Quelques sugg	109
870 POKE 36,52: PRINT M;" " ;TR\$;	DBF3	estions"	D692
OBS\$;:OBS\$ = "": GOTO 880		975 VTAB PEEK (37) + 3: HTAB 20	
875 POKE 36,52: PRINT OBS\$;:OBS\$	E2AA	: PRINT "Répondre par le num	ED71
= "": PRINT SPC( 60);AJ\$;:A	75BA	éro convenable ";	110
J\$ = ""	63B1	980 GET C\$: PRINT D\$: IF C\$ > B\$	110
880 PRINT		OR C\$ < "1" THEN VTAB PEEK	111
885 RETURN		(37): POKE 36,53: GOTO 980	87F8
890 REM PASSAGE D'UN HEMISPHERE		985 RETURN	63B1
A L'AUTRE		990 REM HEURE DE DEPART	
895 OBS\$ = "Au delà du pole: fus		995 VTAB 12: HTAB 20: PRINT "Heu	111
eau":AJ\$ = "opposé (12 H d'é	DE66	re = " ;: POKE 36,40: PRINT "	112
cart)		Minutes = "	3C66
900 K = A:A = A + 12: IF K > 12	4AC3	1000 GOSUB 595	FA53
THEN A = K - 12	63B1	1005 H = P + PM / 60:H1 = P:H2 =	40F2
905 RETURN		PM	63B1
910 HOME : VTAB 5: HTAB 25: PRIN		1010 RETURN	
T "Vous vous trouvez AU POLE		1015 VTAB 23: POKE 36,1: PRINT "D	113
." : PRINT : PRINT : POKE 36,		ans ces conditions il ne bou	E4EA
57: PRINT "^^": PRINT "1" Il		ge pas !": GOSUB 685: RETURN	114
n'y a qu'une seule direction		1020 REM CALCUL DE L'HEURE	114
possible : "le sud au pole n	B9EF	1025 HOME : HTAB 29: PRINT "CALCU	115
ord,"	3A51	L DE L'HEURE"	DB73
915 POKE 36,58: PRINT "^^"	6914	1030 VTAB 6	2CD8
920 POKE 36,46: PRINT "le nord a		1035 PRINT "Nous allons calculer	
u pole sud."		l'heure qu'il est dans un en	116
925 PRINT : POKE 36,28: PRINT "^^		droit donné par rapport à ce	116
": POKE 36,40: PRINT "^^":		lle"	E269
POKE 36,54: PRINT "^^"	7EC1	1040 PRINT "que vous attribuez à	117
930 PRINT "2" Il n'y a pas d'heu		la France."	D33A
re au pole, ou plutot il est		1045 PRINT	75BA
en meme temps toutes les "		1050 PRINT : PRINT "Il s'agit de	
		l'HEURE DU FUSEAU. L'heure l	118
		égale peut en différer, ains	118
		i en France"	66C7
			119

1055 PRINT "où nous sommes en avance sur le soleil de 2 heures en été et de 1 en hiver."	D08C	1195 PRINT H2;" m ";DTE\$;". "	775E
1060 PRINT	758A	1200 B\$ = "4"	9869
1065 PRINT : PRINT "Ces aménagements artificiels d'horaire étant très variables dans le monde, nous	E396	1205 GOSUB 685: GOSUB 945: IF C\$ = "3" THEN CLEAR : GOTO 130	286E
1070 PRINT : PRINT "n'allons pas en tenir compte."	229F	1210 IF C\$ = "2" THEN CLEAR : GOTO 1080	E789
1075 GOSUB 685	0B53	1215 IF C\$ = "4" THEN GOTO 555	C06A
1080 HOME : VTAB 10:Y = 180:Z2 = 12	ABC9	1220 W = 1:B\$ = "2": HOME : VTAB 12: HTAB 15	6184
1085 PRINT "Indiquer la longitude dont vous voulez connaître l'heure": GOSUB 590	96F5	1225 PRINT "Voulez-vous changer 1 = La longitude": PRINT : POKE 36,34: PRINT "2 = L'heure en France": GOSUB 975	5962
1090 LG = (P * 60 + PM):L1 = P:L2 = PM	7695	1230 IF C\$ = "1" THEN GOTO 1080	F991
1095 PRINT : PRINT : HTAB 6: PRINT "Est ou Ouest ? (taper E ou O) ";; GET O\$: IF O\$ < > "E" AND O\$ < > "e" AND O\$ < > "O" AND O\$ < > "o" THEN VTAB 12: GOTO 1095	E745	1235 GOTO 1110	486E
1100 PRINT O\$	852D	1240 REM CALCUL DE LA LONGUEUR D'UN PARALLELE ET DE LA LARGEUR D'UN FUSEAU	
1105 IF W = 1 THEN GOTO 1115	A63C	1245 Z2 = 12	01BF
1110 HOME : VTAB 10: PRINT "Quelle heure est-il en France ? " :Y = 24: GOSUB 990	AC91	1250 HOME : GOSUB 570	3C1D
1115 IF LG - 450 < 0 THEN H = 0	9BAF	1255 IF P = 90 THEN PM = 0	5497
1120 H = INT ((LG - 450) / 900) + 1	0BDF	1260 LT = (P + PM / 60) * PI / 180:L1 = P:L2 = PM	0A6A
1125 P2 = H1 + H: IF O\$ = "o" OR O\$ = "O" THEN P2 = H1 - H	F2DC	1265 LP = 40000 * COS (LT):LF = LP / 24	E562
1130 DTE\$ = "Ce jour": IF P2 > 24 THEN P2 = P2 - 24:DTE\$ = "le lendemain"	CCEF	1270 LP = INT (LP * 100) / 100:LF = INT (LF * 100) / 100	9FEC
1135 IF P2 < 0 THEN P2 = 24 + P2: DTE\$ = "la veille"	25B3	1275 HOME : VTAB 10: HTAB 15	9A0A
1140 PRINT	758A	1280 PRINT "Longueur du parallèle ";L1;" " ";L2;"' = ";LP;" Kilomètres"	736D
1145 HOME : VTAB 10: POKE 36,20	3CBE	1285 VTAB 13: HTAB 15	553C
1150 PRINT "France":; POKE 36,49: IF L1 < 10 THEN PRINT " "	7E90	1290 PRINT "Largeur d'un fuseau à cette latitude :";LF;" Kilomètres"	2DB5
1155 PRINT L1;" " ";: IF L2 < 10 THEN PRINT "0":	8C30	1295 VTAB 16	5E09
1160 PRINT L2;"' " ";: PRINT O\$	36C0	1300 HTAB 15: PRINT LF" Kilomètres est aussi la vitesse à laquelle se déplace"	7C8F
1165 PRINT : POKE 36,18	71AB	1305 HTAB 15: PRINT "un point "; CHR\$ (34);"immobile"; CHR\$ (34);", en raison de la rotation de la terre"	504A
1170 IF H1 < 10 THEN PRINT " ";	E575	1310 VTAB 22: PRINT " Pour revenir au menu, frapper ESCAPE"	0717
1175 PRINT H1;" H "":; IF H2 < 10 THEN PRINT "0":	CC55	1315 GOSUB 685	0B53
1180 PRINT H2;" m":	D35B	1320 IF C\$ = CHR\$ (27) THEN GOTO 130	3388
1185 POKE 36,45: IF P2 < 10 THEN PRINT " ";;	12A9	1325 GOTO 1240	6B72
1190 PRINT P2;" H "":; IF H2 < 10		1330 GOTO 130	173F
		1335 REM HEURE D'UN LIEU DU TABLEAU	

(suite page 8)



# HEURES

(suite)

```

1340 Z = 22:Y = 40          6287
1345 HOME                  2F97
1350 RESTORE               5DAE
1355 FOR I = 1 TO 40      62F0
1360 READ A$: READ A      ADEE
1365 IF I > 20 THEN VTAB I - 20:
      POKE 36,40          2DED
1370 IF I < 10 THEN PRINT " "; 7E45
1375 PRINT I;" ";        2BFD
1380 PRINT A$             D11F
1385 NEXT I               9DCB
1390 VTAB 22:Z = 67: PRINT "Quel
      numéro choisissez-vous ? (po
      ur revenir au menu, frapper
      99 )";: GOSUB 700   2C36
1395 IF X = 99 THEN GOTO 130 6D4A
1400 IF X < 1 OR X > 40 THEN VTA
      B 23: POKE 36,1: PRINT "Donn
      er un numéro entre 1 et 40 !
      ": GOSUB 685: VTAB 22: PRINT
      CHR$(11): GOTO 1390 5F4B
1405 N = X                 1076
1410 HOME : VTAB 10:Y = 24:Z = 1
      2                   7D96
1415 PRINT "Quelle heure est-il e
      n France ? ": GOSUB 990 D3BA
1420 RESTORE              5DAE
1425 FOR I = 1 TO N       E1DA
1430 READ A$: READ A      ADEE
1435 NEXT I               9DCB
1440 DTE$ = "ce jour"     C7FD
1445 H = H1 + A: IF H > 24 THEN H
      = H - 24:DTE$ = "le lendemai
      n"                  63DE
1450 IF H < 0 THEN H = 24 + H:DTE
      $ = "la veille"     0A05
1455 HOME : VTAB 12: HTAB 20 9E08
1460 PRINT "S'il est ";   B974
1465 IF H1 < 10 THEN PRINT " "; E575
1470 PRINT H1;" H ":: IF H2 < 10
      THEN PRINT " ";    3025
1475 PRINT H2;" m en FRANCE," 5ACE
1480 PRINT : HTAB 20: PRINT "il e
      st """;            7760
1485 IF H < 10 THEN PRINT " "; 2744
1490 PRINT H;" H ":: IF H2 < 10 T
      HEN PRINT " ";    12F4
1495 PRINT H2;" ";DTE$;" à ";A$ B7AE
1500 GOSUB 685           0B53
1505 GOTO 1335           5577
1510 REM SUGGESTIONS

```

```

1515 HOME                  2F97
1520 PRINT "1") Se placer à la la
      titude 90": PRINT 7148
1525 PRINT "2") Heure 12, vitesse
      1000, distance 40000 Km, se
      placer successivement aux" A29D
1530 HTAB 5: PRINT "latitudes 0,6
      0,89": PRINT       4025
1535 PRINT "3") Latitude 0, Heure
      12, vitesse 1666.666 Km, fa
      ire varier les distances et" 3ACA
1540 HTAB 5: PRINT "constater ce
      qui se passe lorsque l'on va
      vers l'ouest."     ADA4
1545 HTAB 5: PRINT "Reproduire le
      phénomène pour d'autres lat
      itudes : "vitesse = largeur d
      u": HTAB 5: PRINT "fuseau, c
      alculée en exécutant le 4" d
      u menu"            6A49
1550 PRINT : PRINT "4") Latitude
      0, Heure 0 H 30, vitesse 200
      0, distance 6000 : "voir ce q
      ui se passe";: HTAB 5: PRINT
      "quand on va vers l'ouest. F
      aire ensuite varier les donn
      ées."              2760
1555 PRINT : PRINT "5") Latitude
      0, Heure 12, vitesse 1000 Km
      , distance 10001. Se placer
      ensuite à": HTAB 5: PRINT "u
      ne autre latitude." 603A
1560 PRINT : PRINT "6") Latitude
      0, Heure 12, vitesse 1000, d
      istance 20001. Constater ce
      qui se": HTAB 5: PRINT "pass
      e quand on va vers l'est ou
      l'ouest"          E2B6
1565 PRINT : PRINT "7") On peut a
      bandonner le schéma de l'avi
      on : "le programme s'applique
      aussi": HTAB 5: PRINT "bien
      au piéton qu'à la fusée."; 36ED
1570 GOTO 485            294C

```

Ne cherchez plus nos courtes routines en langage machine : la plupart d'entre elles sont regroupées (avec d'autres, inédites), dans des recueils pratiques, accompagnés d'une disquette.

Bulletin de commande page 75.

# Transfert de variables du Basic vers un programme en assembleur

La programmation en Assembleur permet d'écrire des programmes plus courts et surtout plus rapides que le Basic. Les revues consacrées à l'Apple publient de nombreuses routines prêtes à être incorporées à vos programmes. Ces applications nécessitent souvent le transfert de données en provenance du programme Basic appelant. La méthode la plus courante, que je trouve personnellement peu conviviale, est de poker les différentes valeurs à des adresses libres en mémoire (généralement dans les trous de la page 0) où le programme Assembleur pourra les retrouver. Je pense qu'il est préférable de faire l'appel par un **CALL adresse, variable1, variable2, ... variable n**, d'autant que les routines de la ROM sont disposées à vous faciliter la tâche. Ce sont ces routines que je vous propose de découvrir et d'utiliser pour le transfert de variables entières.

## 1. Un nombre entier entre 0 et 255 : CALL 768,X

\$300 : 20 DE BE JSR CHKCOM ; teste la virgule après l'adresse  
 \$303 : 20 F8 E6 JSR GETBYT ; évalue la variable X  
 \$306 : 86 06 STX 06 ; stocke la valeur en \$06.

GETBYT évalue l'expression. Le résultat est mis dans le FAC et converti en un entier inférieur à 256 et enfin stocké dans X et aussi dans FACLO. La valeur de X est ensuite sauvegardée à l'adresse \$06.

### Il est possible de gagner 3 octets :

\$300 : 20 F5 E6 JSR GETBYTC ; saute un caractère et évalue X. ] GETBYTC saute un caractère avant d'appeler GETBYT.  
 \$303 : 86 06 STX

### Et voici encore une autre possibilité :

\$300 : 20 4C E7 JSR COMBYTE ; teste la virgule et évalue X.  
 \$303 : 86 06 STX

Le sous-programme COMBYTE (\$E74C) vérifie que l'octet pointé par TXTPTR est une virgule, puis appelle GETBYT (\$E6F8).

## 2. Un nombre entier entre 0 et 65535 : CALL 765,X

\$2FD : 20 DE BE JSR CHKCOM ; teste la virgule  
 \$300 : 20 67 DD JSR FRMNUM ; évalue la formule  
 \$303 : 20 52 E7 JSR GETADR ; FAC → LINNUM  
 \$306 : A5 50 LDA 50 ; transfère LINNUM dans \$06  
 \$308 : 85 06 STA 06  
 \$30A : A5 51 LDA 51 ; transfère LINNUM+1 dans \$07.  
 \$30C : 85 07 STA 07

Le nombre sera stocké en \$06 et \$07.

FRMNUM (\$DD67) évalue la formule pointée par TXTPTR (\$B8-B9) en vérifiant que c'est un nombre et range la valeur dans le FAC. GETADR (\$E752) transforme FAC en entier sur deux octets et le range en LINNUM (\$50-\$51).

### 3. Saisie de plusieurs entiers :

CALL 768,E1,E2, ... En E1,E2 ... sont des entiers inférieurs à 256.

\$300 :	20 BE DE	JSR \$DEBE	; teste la virgule
\$303 :	A0 00	LDY £\$00	; initialise le pointeur
\$306 :	84 FA	STY \$FA	; sauvé dans \$FA
\$308 :	20 F8 E6	JSR GETBYT	; évalue la formule
\$30A :	A4 FA	LDY \$FA	
\$30C :	96 06	STX \$06,Y	; stocke la variable
\$30E :	E6 FA	INC \$FA	; incrémente le pointeur
\$310 :	C9 2C	CMP £\$2C	; le code ASCII de la virgule est 2C
\$312 :	D0 06	BNE \$31A	; pas de virgule, c'est fini
\$314 :	20 B1 00	JSR CHRGET	; continue la lecture
\$317 :	18	CLC	; on continue
\$318 :	90 ED	BCC \$308	; la boucle...
\$31A :	... suite du programme.		

Les valeurs seront disponibles en \$06, \$07, \$08, etc. ...

### 4. Si vous préférez les parenthèses, les concepteurs de la ROM de l'Apple ont encore pensé à vous :

CALL 768(X) où X est un entier inférieur à 256 :

\$300 :	20 B2 DE	JSR PARCHK	;
\$303 :	20 FB E6	JSR CONINT	;
\$306 :	86 06	STX \$06	

PARCHK (\$DEB2) vérifie la présence d'une parenthèse ouvrante, évalue la formule qui suit par FRMEVL (\$DD7B) et teste la présence de la parenthèse fermante. La valeur du FAC est transformée en entier dans X par CONINT (\$E6FB).

#### On pourrait aussi utiliser :

\$300 :	20 F5 E6	JSR GETBYTC	
\$303 :	20 B8 DE	JSR CHKCLS	; teste la parenthèse fermante.
\$306 :	86 06	STX \$06	

# APPLE // ProDOS

## GUIDE DU PROGRAMMEUR APPLESOFT

Par Marcel COTTINI

Vous ne tirerez pleinement profit de votre Apple que si vous connaissez parfaitement les multiples ressources de son nouveau système d'exploitation : ProDOS.

Après avoir lu l'excellent ouvrage de Marcel COTTINI, vous trouverez en ProDOS l'un de vos meilleurs alliés pour une programmation avancée.

#### EXTRAIT DE LA TABLE DES MATIÈRES :

- Familiarisation du lecteur avec ProDOS
- Les vecteurs particuliers sous ProDOS
- Système DOS 3.3 ou ProDOS ?
- Monitor, interpréteur Applesoft et ProDOS
- Le boot d'une disquette ProDOS
- Occupation type d'une configuration Apple
- Commandes typiques du Basic.System
- La Basic System Global page
- et bien d'autres sujets ! BULLETIN DE COMMANDE PAGE 75.

# Votre bibliothèque INFORMATIQUE

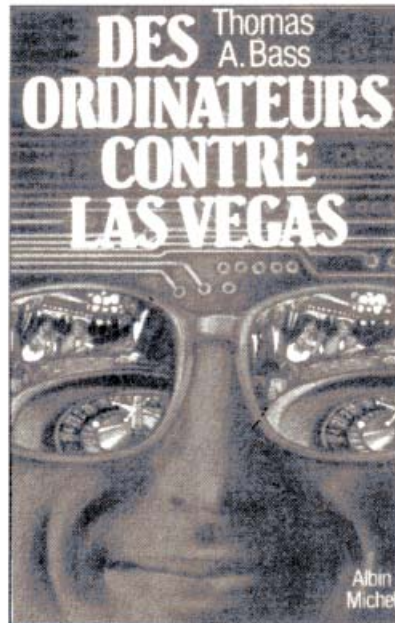
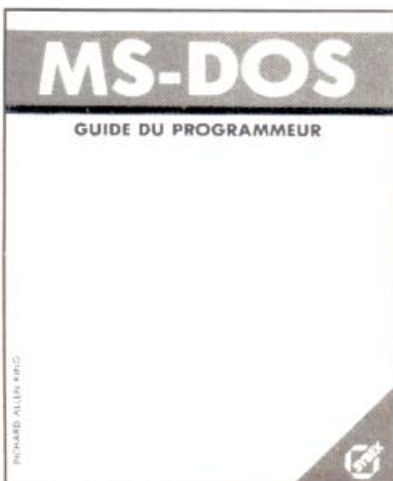
par **NESTOR**

- **MS-DOS, GUIDE  
DU PROGRAMMEUR**  
(Richard Allen King)

Rassurez-vous, applemaniques mes amis, je ne suis pas vendu au standard IBM. Mais je ne suis pas non plus — malgré mon grand âge — l'un de ces êtres rétrogrades qui refusent le progrès. Le standard en question existe. MS-DOS est actuellement le système d'exploitation le plus utilisé sur les micro-ordinateurs 16 bits professionnels. Il est bon de savoir à quoi il ressemble... et je vous prie de croire que l'on n'est pas déçu. Le guide de Richard Allen King (traduction française de Claire Baccichet) présente toutes les informations nécessaires à l'utilisateur et... au programmeur (il n'est pas inutile de faire la distinction !).

La première partie décrit le fonctionnement de MS-DOS versions 2 et 3, la structure des disques et des fichiers, les fonctions du BDOS, la gestion du clavier, de l'écran et des interfaces séries et parallèles. La deuxième partie s'adresse plutôt au lecteur non programmeur (il en existe beaucoup du côté des compatibles, on le sait). On y trouve les détails de l'utilisation de l'éditeur Edlin, des fichiers batch et du programme de mise au point Debug, ainsi que de nombreux utilitaires.

SYBEX, 6-8 Impasse du Curé, 75881 PARIS CEDEX 18  
416 pages — 248 F TTC.



- **DES ORDINATEURS  
CONTRE LAS VEGAS**  
(Thomas A. Bass)

Des doutes m'assaillent quand on précise qu'un récit, aussi extravagant que faire se peut, est pourtant l'étrange et authentique aventure d'une bande de physiciens et d'informaticiens sorciers à l'assaut des casinos...

Il paraît que tout cela est vrai et que cette passionnante histoire de 350 pages n'est pas un roman. Comme vous allez évidemment vous offrir ce bouquin, je suppose que nous ne tarderons pas à entendre parler de vous... lorsque vous aurez, grâce à un micro-ordinateur glissé dans la semelle de vos chaussures, ruiné tous les casinos de France et de Navarre !

Attention ! les choses ne seront pas si simples. Vous devrez non seulement réinventer toutes les procédures mises au point par la petite bande, mais apprendre à pianoter avec vos orteils. Evidemment, vous bénéficierez de l'expérience de nos héros, mais vous partirez néanmoins avec un réel handicap : nous ne sommes pas à Las Vegas,

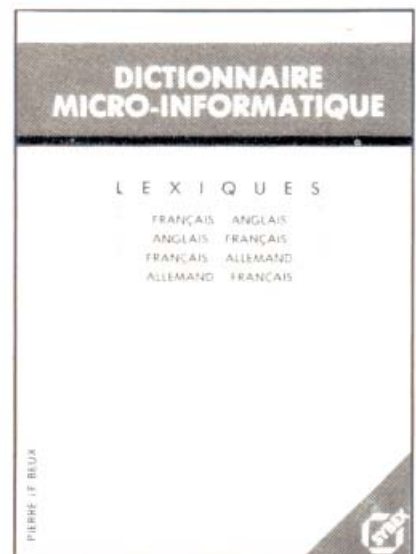
mais en France... et vous n'êtes pas Américain !

Editions Albin Michel,  
22, rue Huyghens, 75014 PARIS  
350 pages — 95 F TTC.

- **DICTIONNAIRE  
MICRO-INFORMATIQUE**  
(Pierre Le Beux)

Un dictionnaire de plus... Notez que l'évolution perpétuelle de la micro-informatique et de son vocabulaire justifie à elle seule les multiples éditions qui tentent de nous faire mieux connaître les mots employés dans les ouvrages techniques. Celui-ci présente l'avantage de comporter des lexiques français-anglais et anglais-français (ce qui n'est pas vraiment original), mais aussi français-allemand et allemand-français, ce qui est moins courant et séduira probablement de nouvelles catégories de lectrices et lecteurs. Vous constaterez que nos voisins allemands empruntent volontiers les termes anglais. A tort ou à raison. Que cela ne nous empêche pas d'exiger, en France, des documentations rédigées en Français, et non en franglais.

SYBEX, 6-8 Impasse du Curé, 75881 PARIS CEDEX 18  
200 pages — 98 F TTC.



Fonctionne aussi sur l'APPLE IIGS

# VIDEC

## EFFACEMENT PROGRESSIF DE L'ÉCRAN

Cette routine fonctionne aussi bien en 40 qu'en 80 colonnes. Son principe est simple. Chaque caractère est lu et sa valeur décrétementée de 1 jusqu'au moment où elle est égale à l'espace (\$A0).

```

100 TEXT : NORMAL : D$ = CHR$ (4) : PRINT D$"PR&3" : PRINT : HOME
105 HTAB 9 : INVERSE : PRINT " VIDEC (EFFACEMENT PROGRESSIF DE L'ÉCRAN EN 40 OU 80 COLONNES) " : NORMAL
110 POKE 34,2 : VTAB 3
115 FOR I = 768 TO 823 : READ R : POKE I,R : NEXT
120 DATA 169,0,133,6,162,23,138,32,71,248,160,39,32,40,3,44,31,192,16,9,44,85,192,32,40,3,44,84,192,136,16,236,202,
16,227,165,6,208,217,96,177,38,201,160,240,9,56,233,1,145,38,169,1,133,6,96
125 LIST : LIST : N = 80 : GOSUB 145
130 PRINT CHR$ (17) : N = 40 : HOME : LIST : GOSUB 145
135 PRINT CHR$ (18) : VTAB 22 : PRINT "(M)ENU DE DISQUETTE " : GET R$ : IF R$ = "m" OR R$ = "M" THEN PRINT D$"RUN MENU"
140 HOME : END
145 PRINT : VTAB 24 : INVERSE : PRINT "ESSAI EN "N"COLONNES" : NORMAL : CALL - 198 : POKE 49168,0 : WAIT 49152,128 : POKE 49168,0 : CALL 768 : RETURN
    
```

300 :	A9 00	LDA £\$00	]	On initialise cette adresse-test (on peut prendre n'importe quelle adresse disponible).
302 :	85 06	STA \$06	]	
304 :	A2 17	LDX £\$17	]	X va servir de compteur pour le nombre de lignes.
306 :	8A	TXA	]	On le passe dans A pour GBASCALC qui va nous placer l'adresse de base de la ligne en \$26-27.
307 :	20 47 F8	JSR \$F847	]	Y = 39 (on va lire de 39 à 0 inclus).
30A :	A0 27	LDY £\$27	]	Vers le sous programme.
30C :	20 28 03	JSR \$0328	]	
30F :	2C 1F C0	BIT \$C01F	]	Si on n'est pas en 80 colonnes, saut.
312 :	10 09	BPL \$031D	]	
314 :	2C 55 C0	BIT \$C055	]	
317 :	20 28 03	JSR \$0328	]	Sinon même chose en AUX pour les 80 colonnes.
31A :	2C 54 C0	BIT \$C054	]	
31D :	88	DEY	]	
31E :	10 EC	BPL \$030C	]	On boucle jusqu'à Y = 0 inclus.
320 :	CA	DEX	]	
321 :	10 E3	BPL \$0306	]	Idem pour le nombre de lignes.
323 :	A5 06	LDA \$06	]	Si \$6 contient autre chose que le 0 initial, encore un tour complet d'écran !
325 :	D0 D9	BNE \$0300	]	Retour au BASIC.
327 :	60	RTS	]	

**Sous-PROG.**

328 :	B1 26	LDA (\$26),Y	]	Lecture du caractère.
32A :	C9 A0	CMP £\$A0	]	Si c'est un espace, rien à faire.
32C :	F0 07	BEQ \$0335	]	
32E :	38	SEC	]	
32F :	E9 01	SBC £\$01	]	Sinon on le décrémente de 1 et on l'affiche.
331 :	91 26	STA (\$26),Y	]	
333 :	A9 01	LDA £\$01	]	On met 1 dans l'adresse-test.
335 :	85 06	STA \$06	]	
337 :	60	RTS	]	Retour.

BSAVE VIDEC.LM,A\$300,L\$38

Le l

Je ce  
gens  
ver "  
les fe  
n'est  
refai  
taire  
don  
affir  
quelc  
né de

Ne n  
vaise  
pens  
nous  
bles  
grâc

Bien  
revie  
bilité  
soft  
TOO

Que  
l'ass  
nou  
teur:  
négl  
Basi  
teur,  
sera

Le s

Tene  
ceux  
fianc  
infor  
près  
Parf  
chev  
teur:

TRE

# C TRÈS SIMPLE !

## Apprenez à programmer en C sur le GS

### Le IIGS la déception ou l'état de grâce

Je connais quelques déçus de l'APPLE IIGS, des gens comme vous et moi, qui s'attendaient à trouver "LE" super basic MICROSOFT gérant la souris et les fenêtres encore mieux que ne le fait "MINIE" (ce n'est pas peu dire !), et puis voilà : Apple nous refait le coup de l'APPLESOFT pur, dur et réglementaire. Mon ami et complice Yvan Kœnig, avec son don naturel du sacrifice et du paradoxe, a beau affirmer "Less is more", j'entrevois tout de même quelques grimaces parmi les parrains du nouveau-né de la Silicon-Valley.

Ne nous resterait-il plus qu'à faire contre mauvaise fortune bon cœur, et à nous consoler en pensant que toutes les petites merveilles que nous avons pu écrire par le passé restent valables et fonctionnent même trois fois plus vite grâce à la WOZ machine ?

Bien sûr que non : se contenter de cette solution reviendrait à n'utiliser que dix pour cent des possibilités de notre nouvelle machine. En effet, l'Apple-soft ne tire pas parti des ROM TOOLS et RAM TOOLS véritable cœur logiciel de l'Apple IIGS.

Que faire ? Je vois des doigts qui se lèvent dans l'assistance... c'est cela même : apprendre un nouveau langage. Aïe ! les trois-quarts des lecteurs de *Tremplin Micro* vont probablement négliger mon article pour aller voir si ça parle Basic un peu plus loin. Amie Lectrice, ami Lecteur, un petit effort s'il vous plaît... et l'avenir sera à vous ! Restez avec nous !

### Le secret informatique

Tenez, je vais confier un secret à celles et à ceux, pleins de courage, qui attendent avec confiance Ma révélation. Je plonge... "LE" secret informatique, le voici : à de rares exceptions près TOUS LES LANGAGES SE RESSEMBLENT !

Parfaitement, on vous a trompés et, à part le fou chevelu qui parle LISP ou PROLOG à des ordinateurs intelligents et le moine bénédictin qui ali-

gne des octets de langage machine au fond d'une cave, les programmeurs utilisant les langages de haut niveau (BASIC, FORTRAN, PASCAL, APL, C etc.), décrivent plus ou moins les mêmes structures et peuvent par conséquent se comprendre...

### Les langages du futur

Ceci étant posé, quel langage choisir ? Une récente prospective américaine laisse entrevoir pour les années 90 la prédominance de trois types :

1. Pour les applications de très haut niveau, commande de fusées, noyaux de langages etc. : survivance de l'assembleur.
2. Pour les applications de haut niveau, systèmes, éditeurs, gestion etc. : langage C.
3. Pour les applications de bas niveau où la vitesse d'exécution n'est pas critique : Basic compilé, genre QUICK-BASIC de l'IBM.

Oublié le PASCAL, désagrégé l'APL, sublimé le FORTRAN : le rouleau compresseur est passé, les spécialistes de l'intelligence artificielle ont devant eux des lendemains qui chantent avec LISP et PROLOG, mais nous autres, pauvres programmeurs sur APPLE, si nous voulons rester crédibles, il va nous falloir sacrifier à la mode et apprendre le langage C.

### C portable

Une bonne nouvelle pour commencer : les programmes que nous écrirons en C pour le GS "tourneront" avec des modifications mineures sur MAC-INTOSH et IBM. C'est ce que l'on appelle la portabilité.

Une deuxième bonne nouvelle : il existe déjà une version professionnelle de C sur le GS. Les petits futés qui ont profités de l'échange "je te donne mon EUROplus plein de poils (et cinq mille cinq cents francs) et tu me donnes ton GS plein de puces" peuvent acquérir, pour une somme modique, une doc tout en anglais pour l'instant,

et une tripotée de disquettes 3,5 pouces comportant entre autres un éditeur de texte, un assembleur version Gééssisé du génial ORCA, un compilateur **C**, un LINKER pour relier tout ça, une bibliothèque bien fournie, un SHELL pas piqué des vers, un ...

Hé ! Hé ! il faut tout ça pour programmer en **C** ! et vous avez le toupet de dire **C** très simple !

Mais oui, braves gens comblés par tant de bons logiciels, il faut tout ça, mais ces outils sont là pour vous simplifier la tâche ou pour améliorer la qualité de votre travail.

J'entends, dans le fond de la salle, des individus murmurer que sur leur APPLE IIe, avec leur BASIC, quand on tape PRINT "HELLO" l'ordinateur répond bien gentiment HELLO sans faire de chichi, mais aussi sans LINKER et autres babioles logicielles... J'avoue volontiers qu'un environnement de type professionnel présente évidemment des inconvénients, et j'aurai bien souvent l'occasion de vitupérer le système de développement APPLE. Toutefois, malgré sa lourdeur, c'est le seul outil qui permette de tout faire et de surcroît de le faire bien : pensez qu'un programme en **C** tourne de 20 à 40 fois plus vite que son homologue en BASIC !

Pour les frais émoulus du Basic, je vais présenter rapidement le système APPLE PROGRAMMER'S WORKSHOP (dans sa version 1.0A7, si l'on en croit la notice technique fournie avec la bête).

### LE SHELL

Première surprise : une fois la disquette APW **C** chargée, on ne retrouve pas le "paysage" PRODOS habituel ; l'écran s'efface et le PROMPT (le caractère qui précède le curseur) n'est pas le \$ habituel, mais le £. En haut de l'écran, trois lignes nous apprennent que nous venons de charger APW SHELL.

En fait, le SHELL est une interface logicielle qui fournit des commandes simplifiant l'écriture de programmes en différents langages. Pour connaître la liste de ces commandes, tapons HELP. Aussitôt, l'écran se remplit aux deux tiers de mots mystérieux : CMPL, CMPLG, COMPILE, LINK et d'autres à consonance plus familière, tels que CATALOG, EDIT, COPY, PREFIX, DELETE, EXEC, etc. Pas d'affolement donc, d'autant moins que, si on tape HELP mot-inconnu (par exemple HELP CMPL), le GS nous explique (dans la langue de

WOZNIAK) la fonction et le mode d'appel du mot mystérieux.

Au cours de notre initiation, nous allons rencontrer trois familles de commandes :

- Les aides documentaires : HELP, CATALOG
- Les aides de programmation : EDIT, COMPILE, LINK, COPY, DELETE
- Les commandes de configuration : EXEC, TEXT

Rassurez-vous, nous n'utiliserons pas les 54 commandes disponibles, celles que je viens de citer suffisant dans 99% des cas.

### UTILISONS LE SHELL

Avant de nous lancer dans la programmation en **C**, un petit exemple va nous montrer à quel point le SHELL est confortable.

### C SYMPA

Les programmeurs en MSDOS (IBM) ont la fâcheuse habitude de taper DIR pour obtenir la liste des programmes d'une disquette. Bien sûr, PRODOS ne reconnaît pas cette commande et "jette" impitoyablement les infidèles. Le SHELL d'APW ne se comporte pas plus galamment, mais on peut lui apprendre facilement à EXECuter cette nouvelle commande.

Si on tape HELP EXEC depuis le SHELL, le GS nous précise que la commande EXEC permet de créer des programmes de commande. On peut grâce à eux élaborer de nouvelles commandes, enchaînements de commandes SHELL.

Tapons : EXEC  
EDIT DIR

Le disque se met à tourner et, en quelques secondes, nous nous trouvons face à un message d'erreur nous informant que les paramètres de tabulation ne peuvent être trouvés, mais que cela n'a pas de réelle importance puisque l'éditeur travaille avec des valeurs par défaut. On tape RETURN et on se trouve sous l'éditeur de programme. Fort bel EDITEUR au demeurant : 20 pages de la documentation lui sont consacrées et il fait tout ce qu'un honnête homme est en droit d'attendre d'un éditeur plein écran moderne ; 48 commandes en tout, plus 16 macros commandes définissables par l'utilisateur... bref, un vrai plaisir. Pour l'instant nous allons nous contenter de taper la commande SHELL qui remplacera notre DIR préféré : **CATALOG èlè**

Le s  
pass  
UTIL  
Puis

Il ne  
disq  
DIR l  
Un p  
ligne  
de fl

L'uti  
les r  
tard  
linke  
beus  
rons

Puis  
en p  
en l  
Il ne  
que,  
puis  
cice,  
pass  
suiv

Notr  
cons  
mat  
J'ap  
yeu  
sert  
Un p  
sieu  
vent  
fonc  
indie  
cutie  
suivi  
d'ar  
Les

TREA

Le symbole **é** désigne une variable qui sera passée depuis la ligne de commande. Ainsi DIR UTILITIES sera "traduit" par CATALOG UTILITIES.

Puis on tape :

- **CTRL-Q** pour Quitter l'éditeur
- **S** pour Sauver notre "programme" sur disque
- **E** pour retourner au SHELL.

Il ne nous reste plus qu'à taper **DIR**. Miracle ! le disque tourne, le catalogue s'affiche. On essaie DIR UTILITIES... OK ça MARCHE !

Un petit raffinement nous permet de rappeler les lignes précédemment tapées sous SHELL à l'aide de flèche HAUTE et de flèche BASSE.

L'utilité de mon exemple n'est pas flagrante pour les non habitués d'IBM, mais nous verrons plus tard l'importance d'un tel outil : compilateur et linker nécessitent l'emploi d'une syntaxe verbuse et complexe que grâce à EXEC, nous pourrions ramener à la frappe d'une lettre.

### LE COMPILATEUR

Puisque nous savons utiliser l'éditeur nous allons en profiter pour taper notre premier programme en langage **C** et observer ce qui se passe.

Il nous faut taper **CC** pour informer le SHELL que, dorénavant nous allons travailler en **C**, puis **EDIT EX01.C** (nom de notre premier exercice, n'oubliez pas le **.C**). Lorsque nous sommes passés en mode d'édition nous entrons les lignes suivantes :

```
main()
é
printf("HELLO GS GOOD-BYE APPLE llçn");
è
```

Notre sympathique machine vibrera de plaisir en constatant que nous parlons sa langue maternelle...

J'aperçois des BASICCOIS qui me font les gros yeux : pas de numéro de ligne ? et non, car ça ne sert à rien.

Un programme en **C** est composé d'une ou plusieurs fonctions décrivant les opérations qui doivent être effectuées. Ici une seule fonction, la fonction **main()**. Ce nom réservé et obligatoire, indique au compilateur où doit commencer l'exécution du programme. Les parenthèses vides qui suivent le nom de la fonction signalent l'absence d'argument.

Les caractères **é** et **è** délimitent un bloc d'instruc-

tions (en fait ces caractères français correspondent aux accolades "curly brackets" de l'anglais).

Dans notre cas, le bloc d'instruction se ramène à une seule fonction : **printf**, fonction de la bibliothèque **C** et permettant d'écrire sur le terminal.

La chaîne de caractères **Hello GS Good-Bye APPLE ll** se termine par **çn**, ce qui correspond à l'envoi d'un retour chariot (carriage return), tandis que le ; est un séparateur d'instruction proche parent du : du Basic.

Votre curiosité justifiée sur la syntaxe de notre programme étant satisfaite, il ne nous reste plus qu'à demander à la machine de l'exécuter. Quittons l'éditeur de la façon habituelle (**CTRL-Q S E**) et tapons : **CMP LG EX01.C KEEP = EX01**

Ce qui signifie en argot GS : **COMPILE** et **LINK** le code source **EX01.C**, conserve le code exécutable généré sous le nom de **EX01**, puis exécute **EX01**.

La disquette se met à tourner et au bout d'un certain temps (plutôt long), le GS nous informe que le compilateur est chargé et qu'il s'occupe de **EX01.C**. Un temps certain, et on ajoute que la compilation est terminée et que le **LINKER** en est à sa première passe.

Une deuxième passe et c'est fini : le programme se charge et s'exécute, devant nos yeux éblouis, les mots magiques **Hello GS Good-Bye APPLE ll** viennent de s'écrire en blanc sur fond bleu.

Ouf ! le temps d'écrire à notre vieille mère que nous parlons le **LANGAGE C**, de faire fondre une plaque de bronze "Programmeur Professionnel" pour mettre devant notre porte et on essaie de comprendre ce qui s'est passé.

### COMMENT C

Je vois des individus (toujours les mêmes) qui prétendent qu'en **BASIC APPLESOFT**...

C'est la dernière fois que j'admets une interruption : il ne faut comparer que des choses comparables : **BASIC APPLESOFT** est interprété, c'est-à-dire que quelque part, caché dans des ROMS, se vautre un gros programme qui vient traduire mot à mot, et au fur et à mesure de leur exécution, les lignes de votre programme. **C** est au contraire compilé : la traduction est faite une fois pour toute en langage machine par un esclave docile : le compilateur. *(suite page 16)*



Le compilateur génère un code (le code objet) contenant la traduction de votre programme et faisant appel à des sous-programmes extérieurs (dans notre cas la fonction **PRINTF**) ; le **LINKER** (éditeur de liens en français) crée un code **EXE**-cutable en liant le code objet et des modules extraits d'une **LIBRAIRIE**.

Une telle méthode est contraignante au niveau du développement : chaque fois que l'on changera une instruction il faudra recompiler et relinker le programme, mais le gain en rapidité d'exécution est considérable. De plus, le programme source est entièrement contrôlé par le compilateur (notons qu'en **C** le contrôle syntaxique est plutôt réduit).

Autre avantage, les utilisateurs finaux n'ont pas accès au listing de votre programme et ne peuvent donc pas y effectuer des modifications dangereuses.

L'utilisation d'un **LINKER** vous permet également de lier des programmes écrits dans des langages différents : **C**, **PASCAL** ou **ASSEMBLEUR** et de construire un programme par petits modules que vous **LINKER**ez à la fin.

## QUAND C OU C ?

Le langage **C** n'est pas une nouveauté : Dennis **RITCHIE** a écrit le premier compilateur en 1972 (le **BASIC** date de 1964).

Au départ, **C** devait faciliter l'écriture des compilateurs ou des systèmes d'exploitation d'ordinateurs puissants genre **PDP 11** de **DIGITAL** (**UNIX** est pratiquement entièrement écrit en **C**), mais son implantation sur les petits systèmes 16 bits l'a rendu extrêmement populaire parmi les développeurs **IBM** ou **MACINTOSH**. Il ne fait pas de doute que son introduction sur la nouvelle génération **APPLE IIGS** va renouveler le type d'applications disponibles.

Un ou deux compilateurs existent également sur l'**APPLE II** (**AZTEC**), mais leur intérêt est surtout d'ordre pédagogique. Ne vous précipitez pas pour acheter la version actuelle du compilateur : la prochaine sera sans doute plus "**USER FRIENDLY**".

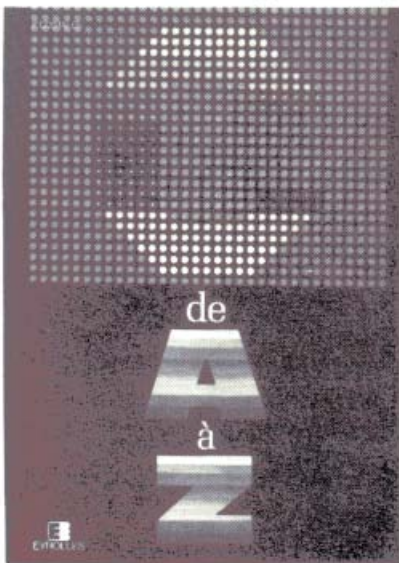
Rongez votre frein, nous allons en rester là pour cette fois. Mon prochain article décrira le langage **C** du point de vue des objets manipulés (variables) et des structures... Bon courage !

**Claude AUBRY.**

# C de A à Z

par Bryan Costales  
EYROLLES éditeur

Nous avons récemment écrit (page 40 du n°13 de *Tremplin Micro*)



tout le bien que nous pensions d'un excellent ouvrage de Robert J. Traister : *DU BASIC AU LANGAGE C* (interéditions).

*C de A à Z*, de B. Costales mérite les mêmes éloges, même si les exemples donnés ne se révèlent pas toujours aussi portables qu'on le souhaiterait... sur le **GS**.

Les bases, conseils, principes généraux de la programmation en **C** restent valables. Aussi en conseillons-nous la lecture (et même l'étude) à celles et à ceux que le **C** intéresse.

**NESTOR.**

## A PROPOS DU LANGAGE C

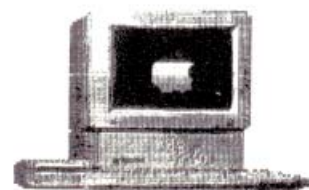
Pratiquer le langage **C** sur un **IBM PC** (ou sur un compatible) est chose relativement facile. La "littérature" est abondante est les programmes en **C** merveilleusement portables (avec les réserves d'usage !).

Avec l'**Apple IIGS**, c'est une autre histoire. La documentation générale est prometteuse, mais en anglais (défaut réhébitorique pour bon nombre de nos compatriotes). De plus, elle ne fournit aucun exemple. Elle se révèle donc lamentable à l'usage.

C'est pourquoi j'approuve sans réserve l'initiative de Guy-Hachette et souhaite qu'il réussisse, grâce au concours de programmeurs (et pédagogues) aussi qualifiés que Claude Aubry et les autres, à décomplexer les **Applemaniaques** et à les orienter vers le **C**.

En attendant le **Super Basic**, **C** : la solution !

**Clément RENARD.**



**GUY-HACHETTE**

# Le bon nombre

## BASIC

Micro-programme  
élémentaire.

```

10 X = 69
20 PRINT "TROUVEZ UN NOMBRE COMPRIS ENTRE 1 ET 100 ->> "
30 INPUT "";N
40 IF N = X THEN PRINT "C'EST LE NOMBRE - TERMINE": END
50 IF N < X THEN PRINT "IL EST TROP PETIT": GOTO 20
60 IF N > X THEN PRINT "IL EST TROP GRAND": GOTO 20

```

## ÉQUIVALENCE EN LANGAGE C

```

main()
é
    int x, n;
    x = 69;
    NB:
        printf("TROUVEZ UN NOMBRE COMPRIS ENTRE 1 ET 100 ->> ");
        scanf("%d", &n);
        if (n == x) é
            printf("C'EST LE NOMBRE - TERMINEçn");
            exit (0);
        è
        if (n < x) é
            printf("IL EST TROP PETITçn");
            goto NB;
        è
        if (n > x)
            printf("IL EST TROP GRANDçn");
            goto NB;
    è

```

**int x,n** Les variables **x** et **n** sont déclarées comme entières.

**NB:** Étiquette utilisée par **goto**.

**printf** Affiche le message.

**scanf** Après **scanf**, il faut un RETURN.

**%d** Indique que la variable à afficher est entière.

**&n** Une variable numérique doit être précédée du signe **&**.

**n == x** Notez la différence entre l'opérateur d'affectation (=) et l'opérateur d'égalité (==).

**n < x**  
**n > x** Utilisation identique à celle du Basic.

**goto NB** Renvoie à l'étiquette NB.

**exit (0)** L'argument 0 n'a généralement pas d'importance.

NE PAS CONSIDÉRER CET EXERCICE COMME UN (BON) EXEMPLE DE PROGRAMMATION EN C !

# Gagnez un Apple, gagnez l'Amérique, devenez auteur Version Soft !

APPLE COMPUTER FRANCE et VERSION SOFT, l'un des plus grands éditeurs de logiciels professionnels sur Apple, lancent un grand concours pour l'attribution des prix : "Les fruits d'une passion 1987".

Cinq prix récompenseront les cinq meilleurs projets de développement sur Apple IIGS ou Macintosh et seront attribués pour la première fois en octobre 1987. Un jury de professionnels sélectionnera les concurrents et désignera les cinq lauréats. Ceux-ci recevront un Apple IIGS ou un Macintosh Plus. De plus, l'auteur du projet le plus original se verra offrir une semaine à Cupertino dans la Silicon Valley, où il sera piloté par Luc Barthelet.

Version Soft pourra proposer aux auteurs des cinq meilleurs logiciels un contrat d'édition et de commercialisation établi selon les usages de la profession.

Auteurs de logiciels, comme Luc Barthelet responsable développement, visez juste, visez la pomme et vivez en Version Soft. Réclamez votre dossier de candidature à : Fabienne Poupard VERSION SOFT 94, rue Lauriston 75116 PARIS — Tél. : (16-1) 47.27.71.72.

## LE RÈGLEMENT

### 1. LES CONCURRENTS

Ce concours est ouvert à toute personne non liée par un contrat d'édition. Il suffit pour participer de renvoyer une fiche d'inscription dûment remplie, accompagnée du dossier de candidature. Les projets de développement peuvent être effectués en équipe. En cas de sélection, le prix sera attribué à son représentant.

### 2. LES PROJETS

Les projets remis à Version Soft peuvent porter sur tous les domaines d'applications, qu'ils soient horizontaux ou sectoriels. Les logiciels peuvent être développés sur Apple IIGS ou Macintosh en langage Pascal, C ou assembleur. Ils devront respecter l'interface Apple (menus déroulants, souris...).

Sans pour autant constituer une liste limitative, les axes suivants sont proposés :

- didacticiels professionnels
- aide à la décision
- structuration des données
- édition électronique
- gestion générale
- gestion sectorielle...

Le jury sera sensible à la créativité, à la puissance et à la convivialité des projets présentés.

### 3. LE DOSSIER DE CANDIDATURE

Le dossier de candidature doit obligatoirement comporter :

- une fiche d'inscription dûment remplie et signée
- un texte de présentation du logiciel (2 pages maximum)
- la liste complète des fonctionnalités
- une disquette de préversion comportant le noyau du programme.

Ce dossier, rempli et complété par les documents demandés ci-contre, en deux exemplaires, est à adresser à : Fabienne Poupard VERSION SOFT, 94, rue Lauriston 75116 PARIS.

### 4. LA DATE LIMITE DE DÉPÔT DES PROJETS

La date limite de dépôt des projets est fixée au mercredi 20 mai 1987 à minuit, le cachet de la poste faisant foi.

### 5. L'EXAMEN DES DOSSIERS

Le jury sélectionnera les vingt meilleurs projets avant le lundi 1<sup>er</sup> juin 1987 et en informera les auteurs. Ces vingt dossiers sélectionnés bénéficieront d'un support technique développeurs Apple et Version Soft, afin de pouvoir poursuivre leur développement jusqu'à la phase finale de sélection des cinq lauréats.

Le 15 septembre 1987 au plus tard, les vingt candidats devront remettre à Version Soft leur programme accompagné d'un synopsis de démonstration. Ne seront pris en compte que les projets qui permettront une utilisation du logiciel par une personne extérieure au développement. Parmi ces vingt logiciels, le jury en sélectionnera cinq, dont les auteurs seront nommés lauréats du concours "Les fruits d'une passion 1987".

### 6. LA REMISE DES PRIX

Les candidats choisis par le jury seront informés par courrier au plus tard le 1<sup>er</sup> octobre 1987. Les prix leurs seront remis officiellement au cours d'une manifestation organisée courant octobre par Version Soft et Apple dans le cadre d'Apple Expo 87.

### 7. LE JURY

Il sera composé de :

— Denis VINCENTI	TF1
— Luc BARTHELET	Version Soft
— Patrick ARNOUX	Challenges
— Yves HEUILLARD	Sciences et Vie Micro
— Neil MINKLEY	Apple Computer France
— Christian ROBERT	Microshop

### 8. LE SUIVI DES LAURÉATS

Un contrat d'auteur sera proposé par Version Soft aux cinq lauréats. Il sera établi selon les usages de la profession. Les lauréats recevront un soutien permanent de Version Soft jusqu'à la commercialisation de leur logiciel. ■

# Quand nos amis belges nous donnent des leçons de logique



**J**'AVOUE avoir été agréablement surpris par la qualité de ce logiciel mis au point par des enseignants et des informaticiens.

Je m'attendais à quelque banalité sans grand intérêt : c'est souvent le cas, dans ce domaine, et pas seulement en Belgique !

L'objectif des auteurs est de faire appel aux facultés de raisonnement de l'enfant, mais l'adulte au cerveau ramolli que je suis, s'est beaucoup amusé avec les locomotives et les wagons de **CONVOI**.

C'est en effet le titre de cette série. L'enfant travaille dans une gare de triage où, à l'aide d'une locomotive — et en s'aidant d'un aiguillage — il doit ranger des wagons numérotés dans un ordre croissant.

Au début de chaque jeu, les wagons sont placés dans un ordre aléatoire. La suite coule de source, mais comme il existe plusieurs niveaux, ce n'est pas toujours aussi simple qu'on pourrait le supposer !

A recommander !

**NESTOR.**

ASBL GRAND-HORNU IMAGES, Rue Sainte-Louise  
B-7320 BOUSSU — Tél. : (065) 33.91.51.

**Configuration nécessaire :**

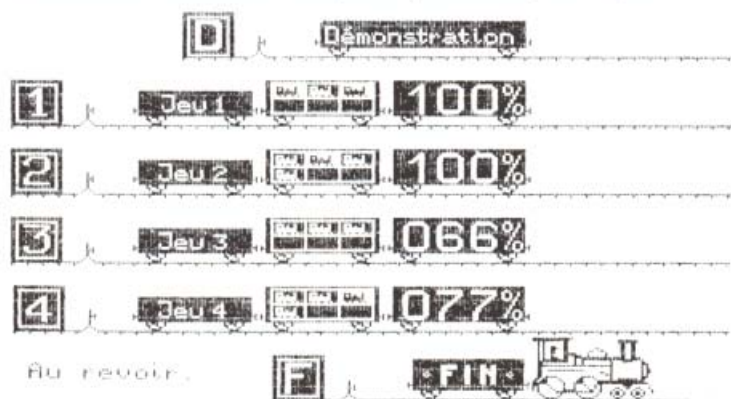
- un micro-ordinateur APPLE modèle II Europlus 48 KRAM minimum, IIe ou IIc ;
- un lecteur de disquettes 5 pouces 140 K octets ;
- un moniteur vidéo monochrome.

**Notes :**

- le moniteur vidéo monochrome peut être remplacé par un moniteur vidéo couleur. Toutefois, le logiciel n'est pas prévu pour utiliser correctement les différentes couleurs ;
- pour imprimer la trace (voir explication plus loin), l'ordinateur doit être connecté à une imprimante (minimum 80 caractères par ligne)



Ces illustrations, extraites du petit livret accompagnant le logiciel, ne donnent qu'une faible idée de la qualité (excellente) des images.



# Votre bibliothèque INFORMATIQUE

par NESTOR

## • **FRAMEWORK I & II FACILE** (Philippe Mercier)

Ce petit guide Marabout vous fera découvrir l'un des best-sellers de l'informatique. Framework, logiciel intégré de réputation mondiale, est à la fois traitement de texte (avec correcteur orthographique), tableur, base de données, logiciel de télécommunication, composeur téléphonique, etc. Les utilisateurs d'IBM et compatibles en savent quelque chose.

Éditions Marabout

192 pages



## • **LA PUCE ET LA PLUME** (Jean Rivier)

Comment se comporte l'écrivain en présence de la Bête ? A-t-il réussi à la domestiquer ? Quels sont les résultats de cette collaboration de l'esprit et du microprocesseur ? Lisez d'abord l'excellente préface de Louis Leprince-Ringuet : elle vous touchera, j'en suis sûr, foi de Nestor ! Et puis, délaissant momentanément votre clavier et vos propres élucubrations, plongez-vous dans celles des autres. J'aurais dû vous le préci-



ser d'entrée : ce livre est une anthologie réunissant les morceaux choisis d'un nombre incroyable d'écrivains. La liste commence par Thérèse de Saint-Phalle et se termine par le nom d'un monument : la Bible, où l'on en évoque un autre, la Tour de Babel, le plus bel exemple d'une communication passant par un langage unique. On imagine mal l'écrivain de demain, armé de sa seule plume !

Le Cherche Midi Editeur,  
68, Rue du Cherche-Midi, 75006 PARIS  
240 pages — 68 F TTC.

## • **CD ROM LE NOUVEAU PAPYRUS** (Bernard Prost)

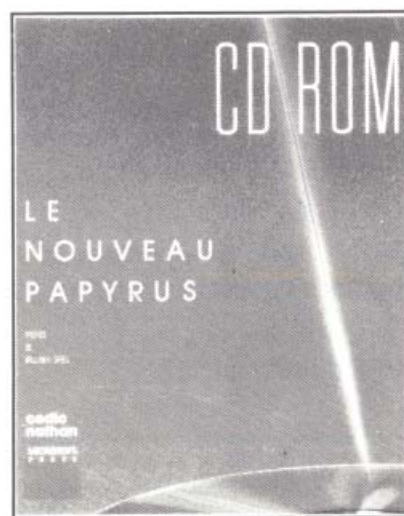
Le compact-disque, cela vous dit quelque chose ? Vous y croyez ? Qu'en attendez-vous sur votre ordinateur personnel ? Sans doute vous posez-vous, comme moi, de multiples questions sur ce nouveau moyen de stockage.

Suivez donc Bernard Prost et faites avec lui le tour de la question. Je vous préviens : ça ne se passera pas sans quelques heures de veille. On n'assimile pas le contenu de 440 pa-

ges en se contentant d'un survol rapide de l'ouvrage. Vous reconnaîtrez que le CD ROM est très différent des autres médias (télévision, cinéma, vidéo, photo, livres, disques et disquettes). C'est en quelque sorte la somme de toutes ces technologies. Les capacités de stockage du CD ROM sont mille fois supérieures à celles de la disquette, pour des coûts pratiquement équivalents. Le livre de Bernard Prost regroupe une somme d'articles dont les auteurs, convaincus des possibilités du compact-disque, sont tous impliqués dans cette nouvelle technologie. Ils sont donc bien placés pour expliquer clairement où en sont les recherches et dans quelles voies elles se dirigent.

A signaler que la première partie de CD ROM est consacrée aux problèmes techniques : système CD, production, éléments de conception, édition sur CD ROM, tandis que la deuxième partie, moins abrupte, aborde les problèmes humains, les projets, le marché et un certain nombre d'applications plus ou moins prioritaires (médecine, droit, archi- vage, recherche, etc.).

Cedic/Nathan — Microsoft-Press  
6, boulevard Jourdan, 75014 PARIS  
440 pages — 280 F TTC.



# PATCH DOS

Lisez d'abord  
le commentaire de Marcel COTTINI

## Configuration requise

Tout Apple II, avec une RAM de 64 Ko ou plus, et le DOS 3.3 booté. Un lecteur de disquettes standard du type Disk II. Pour une exploitation complète du patch, un lecteur de disquettes 35/40/80 pistes est alors requis (type TEAC FD55F par exemple).

## A propos du DOS 3.3 et de ProDOS

Le DOS 3.3 par comparaison à ProDOS, dernier système d'exploitation commercialisé par Apple Computer, demande à être patché pour modifier une de ses commandes ou encore lui adjoindre une commande externe. Le système d'exploitation ProDOS initialise, lors du boot d'une disquette système, deux pages très particulières :

- l'une, la *PRODOS Global Page* aux adresses \$BF00-\$BFFF ;
- l'autre, la *BASIC Global Page* aux adresses \$BE00-\$BEFF.

Cette dernière est particulièrement adaptée pour le programmeur AppleSoft, et autorise par l'intermédiaire du vecteur \$BE06 l'appel d'une commande externe au Basic System. Le fichier BASIC.SYSTEM, interface entre PRODOS et l'interpréteur AppleSoft, contient un certain nombre d'instructions standards implémentées d'origine par le concepteur. A ces commandes internes, le programmeur peut par l'intermédiaire de la *BASIC Global Page*, ajouter une ou plusieurs commandes externes, le système effectuant alors l'exécution de cette commande suivant un protocole établi au préalable. Cette innovation n'est malheureusement pas présente dans le DOS 3.3 qui semble malgré tout très présent dans les réalisations diffusées par les revues spécialisées, *Tremplin Micro* entre autres. Tout au long de sa longue carrière, le DOS 3.3 a connu un certain nombre de modifications. L'apparition de l'Apple IIe (début 1983) a provoqué un bouleversement profond tant au niveau du Soft

que du Hard. Ce fut la vague des patches du DOS 3.3 sous toutes ses formes possibles et imaginables. Certains programmeurs ayant fait l'acquisition de lecteurs de disquettes 35/40/80 pistes restèrent sur leur faim, les programmes utilitaires ayant été largement délaissés. C'est ainsi que prit forme le projet de réaliser mon propre système d'exploitation avec des utilitaires s'y rapportant. Le résultat se concrétisa par un DOS réécrit et réassemblé, en gardant les adresses d'entrées/sorties compatibles au DOS 3.3. Par la suite, certaines idées de base furent réutilisées pour patcher le DOS 3.3.

**PATCH DOS** est un programme en Basic AppleSoft pour ceux qui pratiquent l'Apple II sous le langage AppleSoft, et permet d'apporter au DOS 3.3 certaines améliorations spectaculaires, ne serait-ce qu'une présentation personnalisée du catalogue de la disquette (une version en langage machine sera présentée dans un prochain numéro). Par ailleurs, toute une série de modifications internes seront réalisées, une fois le programme exécuté. Par la suite, toute explication sera fournie pour une meilleure compréhension de certains mécanismes du système d'exploitation.

Ce patch pourra être réalisé avec la plupart des DOS commercialisés, et vous permettra de travailler par la suite avec des disquettes 35, 40 ou 80 pistes, sans aucune précaution préalable. Votre nouveau système est maintenant capable de reconnaître les différents formats.

(suite page 22)

---

## Modifications internes effectuées au DOS 3.3

---

### Keyboard Intercept — \$9D02

*Adresse de la routine d'entrée de caractère.*

A l'adresse \$9D02 se trouve normalement le pointeur de la routine d'entrée de caractère au clavier. Sa valeur pour un DOS 3.3 est \$9E81 (\$9D02- 81 9E). L'auteur a uniquement tenu à respecter la valeur du pointeur pour observer une certaine compatibilité avec d'autres systèmes.

### Video Input Handler — \$9D04

*Adresse de la routine de sortie de caractère.*

A l'adresse \$9D04 se trouve normalement le pointeur de la routine de sortie de caractère. Sa valeur pour un DOS 3.3 est \$9EBD (\$9D04- BD 9E). Comme pour la routine Keyboard Intercept, ce vecteur est maintenu pour respecter une compatibilité avec d'autres systèmes.

### No INIT — \$9D1E

*Elimine la commande INIT.*

A partir de l'adresse \$9D1E débute la table des pointeurs des commandes du DOS. La commande INIT se trouve en tête de la table (\$9D1E- 4E A5). Pour rendre cette commande inefficace, le pointeur de la table est modifié, et se connecte par la suite à l'entrée du démarrage à chaud dans le système. L'entrée DOS WarmStart se trouve à l'adresse \$9DBF. Le nouveau pointeur aura comme valeur : nouvelle adresse -1 (\$9D1E- BE 9D). Une autre modification est rendue nécessaire pour annuler les codes ASCII dans la table nominative des instructions du DOS (\$A884-\$A908). La commande INIT est représentée par les codes ASCII suivants :

A884- 49 4E 49 D4 INIT. Le dernier caractère de chaque instruction a son bit 7 égal à 1, les autres caractères ont le bit 7 égal à zéro.

**Après modification, les caractères ASCII seront les suivants :**

A884- 58 58 58 D8 :::X. Les trois premiers caractères (:) sont neutres, pas de caractère alphabétique, le DOS se trouve ainsi piégé.

Après chaque appel de la commande INIT, un démarrage à chaud sera effectué dans le DOS, au retour un message d'erreur affiché.

### Drive Range ERROR — \$A95B

*Extension du nombre de drives à 4.*

D'origine, le DOS 3.3 limite à deux (00 02) le nombre de drives pouvant être raccordés à une interface contrôleur. Un lecteur de disquettes standard Disk II possède une tête de lecture/écriture unique située en dessous de la disquette.

Un lecteur de la nouvelle génération possède deux têtes de lecture/écriture, un adressage particulier est alors requis. Chaque drive de ce type est vu par le système comme un ensemble de deux lecteurs bien distincts. Equipé d'un tel drive, il vous sera possible d'utiliser l'option D (drive) d'une façon particulière.

**Nouvelles caractéristiques d'un tel drive :**

- Deux drives peuvent être raccordés par interface ;
- Chaque drive sera vu par le système comme un ensemble de deux lecteurs ;
- Le lecteur 1 sera adressé D1 pour solliciter la tête du bas, et D3 pour la tête du haut ;
- Le lecteur 2 sera adressé D2 pour solliciter la tête du bas, et D4 pour la tête du haut.

Cette modification est réalisée au niveau du DOS en modifiant le byte à l'adresse \$A95B (\$A95B-04). Aux adresses \$A955-\$A970 se trouve la table des valeurs des paramètres additionnels valides. Quatre bytes sont réservés pour chaque paramètre : deux bytes pour la valeur minimale et deux bytes pour la valeur maximale. C'est ainsi que les paramètres réservés au drive se trouvent aux adresses \$A959-A95C.

**Valeurs pour un DOS 3.3 :**

\$A959- 01 00 LByte, HByte de la valeur mini (00 01)

\$A95B- 02 00 LByte, HByte de la valeur maxi (00 02)

**Valeurs après modification :**

\$A95B-04 00 LByte, HByte de la valeur maxi (00 04)

### 35/40/80 Tracks — \$BEAF

*Extension du système, traite par la suite des disquettes 35/40 ou 80 pistes.*

Le DOS 3.3 gère par défaut des disquettes formatées 35 pistes, les autres formats n'étant pas reconnus. A partir de l'adressage \$BEAF se trouve la routine de formatage intégrée au DOS 3.3. L'espace mémoire \$BEAF-\$BFB3, réservé normalement pour la routine de la commande INIT, sera remplacé et mis à profit pour y placer une nouvelle routine. Une fois en place, le système reconnaîtra et traitera par

la suite toute disquette avec un format 35/40 ou 80 pistes. Toutes les routines et sous-programmes appelant cette partie du DOS, écriture ou lecture d'un secteur au niveau de la disquette par exemple, seront traitées sans problème.

#### Différents vecteurs annulés de la fonction INIT :

- \$BEAF-\$BF0C Gestion de la commande INIT
- \$BF0D-\$BF61 DISKF2. Routine de formatage de la piste courante.
- \$BF62-\$BF87 NXTSEC. Vérification de la piste formatée.
- \$BF88-\$BFA7 MARKMAP. Mise à jour de la Bit-Map disquette.
- \$BFA8-\$BFB7 SECMAP. Matrice pour marquer la Bit-Map.

#### Allocate Sector Disk — \$B247

Allocation des secteurs d'une disquette.

Modification des adresses \$B247-\$B2B4 pour permettre au système de gérer les pistes comprises entre 00-79 inclus (80 pistes). A ces adresses se trouve la routine d'allocation du numéro du secteur

de la disquette. Après le patch, il sera possible de traiter une disquette 80 pistes, le format 35 ou 45 pistes reste accessible.

#### New VTOC — \$B2D2

Nouvelle VTOC (35, 40 ou 80 pistes).

La table d'occupation d'une disquette standard autorise la gestion de 35 pistes sur une disquette 5 pouces 1/4. A cet effet, certains octets sont réservés dans la Bit-Map, piste \$11 (17) secteur \$00 de la disquette.

#### Volume Table Of Contents — VTOC :

La Vtoc (Volume table Of Contents) ou table d'occupation de la disquette, permet de gérer les secteurs et les pistes de la disquette (Bit-Map), ainsi que l'écriture d'un certain nombre de données et repères utiles pour la lecture et l'écriture d'un secteur. Elle occupe le premier secteur (\$00) de la piste \$11 avec un total de 256 bytes. Les valeurs stockées dans la Vtoc sont en base 16 ou hexadécimal.

(suite page 24)

### Structure d'une Volume Table Of Contents standard

Pos.	Valeur	Contenu	DOS
\$00	\$04	Inutilisé.	\$B3BB
\$01	\$11	Piste du premier secteur/catalogue.	\$B3BC
\$02	\$0F	Numéro du premier secteur du catalogue.	\$B3BD
\$03	\$03	Version du DOS (3 pour un DOS 3.3).	\$B3BE
\$04-\$05		Inutilisés.	
\$06	\$FE	Numéro du volume de la disquette (entre 0 et 254 ou \$00 et \$FE).	\$B3C1
\$07-\$26		Inutilisés.	
\$27	\$7A	122 secteurs maximum dans une liste d'adresses (Tracks Sectors List — TSL) des secteurs occupés par un fichier.	\$B3E2
\$28-\$2F		Inutilisés.	
\$30	\$FF	Dernière piste où des secteurs ont été alloués.	\$B3EB
\$31	\$FF	Direction de l'allocation des pistes (+1 ou -1).	\$B3EC
\$32-\$33		Inutilisés.	
\$34	\$23	Nombre de pistes (\$23 = 35 pistes).	\$B3EF
\$35	\$0F	Nombre de secteurs par piste (16).	\$B3F0
\$36	\$00	Nombre de bytes par secteur = 256 = \$100 (HByte).	\$B3F1
\$37	\$01	(\$36-\$37 octet poids faible et poids fort).	
\$38-\$3B		Occupation de la piste 00 (\$00).	\$B3F3
\$3C-\$3F		Occupation de la piste 01 (\$01).	\$B3F7
\$40-\$43		Occupation de la piste 02 (\$02).	\$B3FB
.....		.....	
.....		.....	
.....		.....	
\$C0-\$C3		Occupation de la piste 34 (\$22).	\$B47B
\$C4-\$FF		Inutilisés, pouvant servir à l'extension de la Bit-Map.	



## Vtbc typique d'une disquette DOS 3.3 standard :

Piste \$11 Secteur \$00

```

00 — 04 11 0F 03 00 00 FE 00
08 — 00 00 00 00 00 00 00 00
10 — 00 00 00 00 00 00 00 00
18 — 00 00 00 00 00 00 00 00
20 — 00 00 00 00 00 00 00 7A
28 — 00 00 00 00 00 00 00 00
30 — 15 01 00 00 23 10 00 01
38 — 00 00 00 00 00 00 00 00
40 — 00 00 00 00 FF FF 00 00
48 — FF FF 00 00 FF FF 00 00
50 — FF FF 00 00 FF FF 00 00
58 — FF FF 00 00 FF FF 00 00
60 — FF FF 00 00 FF FF 00 00
68 — FF FF 00 00 FF FF 00 00
70 — FF FF 00 00 FF FF 00 00
78 — FF FF 00 00 FF FF 00 00
80 — FF FF 00 00 FF FF 00 00
88 — FF FF 00 00 FF FF 00 00
90 — FF FF 00 00 FF FF 00 00
98 — FF FF 00 00 FF FF 00 00
A0 — FF FF 00 00 FF FF 00 00
A8 — FF FF 00 00 FF FF 00 00
B0 — FF FF 00 00 FF FF 00 00
B8 — FF FF 00 00 FF FF 00 00
C0 — FF FF 00 00 00 00 00 00
C8 — 00 00 00 00 00 00 00 00
D0 — 00 00 00 00 00 00 00 00
D8 — 00 00 00 00 00 00 00 00
E0 — 00 00 00 00 00 00 00 00
E8 — 00 00 00 00 00 00 00 00
F0 — 00 00 00 00 00 00 00 00
F8 — 00 00 00 00 00 00 00 00
    
```

Pistes 00 et 01  
 Pistes 02 et 03  
 Pistes 04 et 05  
 Pistes 06 et 07  
 Pistes 08 et 09  
 Pistes 10 et 11  
 Pistes 12 et 13  
 Pistes 14 et 15  
 Pistes 16 et 17  
 Pistes 18 et 19  
 Pistes 20 et 21  
 Pistes 22 et 23  
 Pistes 24 et 25  
 Pistes 26 et 27  
 Pistes 28 et 29  
 Pistes 30 et 31  
 Pistes 32 et 33  
 Pistes 34  
 Inutilisés.

## Occupation des secteurs — Bit-Map :

Initialement, 32 bits étaient réservés pour l'encodage de chaque piste, ce qui représentait l'occupation de 16 secteurs. Uniquement 16 bits sont utilisés, les 16 autres étant simplement ignorés par le DOS 3.3. Les bits représentatifs sont regroupés par ensembles de deux octets, ou  $2 \times 8$  bits. Chaque octet est divisé en demi-octets (quartet ou nibble) pour coder un groupe de 4 secteurs.

## Exemple d'encodage d'une piste — Secteurs \$00-\$0F :

Secteurs : FEDC BA98 7654 3210

Bits : 7654 3210 7654 3210

Valeur possible des bits : soit 1, soit 0.

Chaque bit positionné à 1 représente le secteur concerné comme libre, tandis que chaque bit positionné à 0 indique que le secteur équivalent est occupé. Pour gérer l'occupation des 80 pistes, il sera nécessaire d'étendre la table d'occupation, et d'utiliser les 16 bits ignorés par le DOS 3.3 jusqu'à présent. Le contenu des adresses \$B2D2-\$B2F7 sera modifié à cet effet.

## Table + Face B disquette — \$BD04

*Reconnaissance de la face A ou de la face B (disquette).*

Une disquette standard 35 pistes est lue par la tête de lecture suivant un protocole bien défini : Phase "on" ou "off", mode lecture ou écriture, positionnement du bras de lecture au dessus de la piste et du secteur souhaité, etc.

La modification actuelle autorise par la suite une extension du protocole fixé initialement : la sélection de la tête de lecture/écriture du haut ou du bas (D1, D2 pour le bas ou D3, D4 pour le haut). Les adresses modifiées sont : \$BD04-\$BD4E.

## MYSEEK PARMs — \$BE5A

*Prépare le système avant de solliciter SEEKABS (\$B9A0/).*

Modifie 17 octets (\$BE5A-\$BE69) pour rendre le système compatible suivant la disquette présente (35, 40 ou 80 pistes). Cette routine déplace le bras de lecture au dessus de la piste souhaitée. Elle détermine le nombre de "Phases" par piste et sauvegarde l'information à l'adresse du slot correspondant.

## Recalibrate Disk Head — \$BDD1

*Recalibre la tête de lecture du drive.*

Les adresses \$BDD1-\$BDD6 sont modifiées, et appellent par la suite les sous-programmes correspondant suivant les nouveaux pointeurs mis en place.

## Patch en-tête du catalogue

*Gestion de la commande CATALOG.*

Plusieurs modifications sont effectuées à ce niveau. Le résultat personnalise la présentation du catalogue de la disquette. C'est ainsi que les numéros du volume et du drive, le format de la disquette (35, 40 ou 80 pistes), et le nombre de secteurs libres sont affichés avant la liste des fichiers de la disquette.

## Exemple de présentation d'une disquette :

VOL	DRIVE	TRACKS	FREE
254	001	035	0050

## Modification des différents vecteurs :

\$ADA3-\$ADBD Mise en place du texte de présentation.

\$AE6A-\$AEAF Appel des sous-programmes de l'affichage des valeurs : VOL, DRIVE, TRACKS et FREE.

\$AEBO-\$AF09 Routine de calcul des secteurs libres.  
 \$B3AF-\$B3B5 Initialement réservé au texte : EMULOV KSID. Comporte maintenant deux sauts de ligne.

## Soft utilitaires 35/40/80 pistes

Pour les détenteurs de lecteurs de disquettes capables de traiter des disquettes 35/40 ou 80 pistes sous DOS, l'auteur envisage par la suite la vente par l'intermédiaire de *Tremplin Micro*, d'une disquette utilitaire appropriée.

A cet effet, les amateurs d'un tel Soft devront se faire connaître auprès de *Tremplin Micro* pour une éventuelle souscription de la disquette UTILITAIRE.

Prix de vente conseillé : 150 Francs.

TREMPLIN MICRO — La Petite Motte — Senillé — 86100 CHATELLERAULT

### Contenu de la disquette UTILITAIRE 35/40/80 TRACKS :

- Patch pour modifier le DOS 3.3 ;
- Patch pour modifier le PASCAL (160 pistes) ;
- Patch divers pour accroître la vitesse de traitement suivant le type de drive utilisé ;
- Patch de reconnaissance drive Disk II et TEAC ;
- Patch pour modifier le programme de copie FID ;
- Un programme de copie 35, 40 et 80 pistes ;
- Un programme de formatage 35, 40 et 80 pistes ;
- Conseils d'utilisation.

```

80 TEXT : HOME : VTAB 12: HTAB 12                                B6CC
90 FLASH : PRINT "PATCH EN COURS": NORMAL                      527D
100 REM ctJctJKEYBOARD INTERCEPTctJctJ
110 HEX$ = "9D02:81 9E": GOSUB 520: CALL - 144                  D5E6
120 REM ctJctJVIDEO INPUT HANDLERctJctJ
130 HEX$ = "9D04:BD 9E": GOSUB 520: CALL - 144                  6705
140 REM ctJctJNO INITctJctJ
150 HEX$ = "9D1E:BE 9D N A884:58 58 58 D8": GOSUB 520: CAL
L - 144                                                            80E7
160 REM ctJctJRANGE ERROR DRIVEctJctJ
170 HEX$ = "A95B:04": GOSUB 520: CALL - 144                      A255
180 REM ctJctJ35/40/80 TRACKSctJctJ
190 HEX$ = "BEAF:0A 20 6B BE 4E 78 04 4E 78 04 60 48 68 48
08 A2 00 C9 28 90 02 A2 02 28 7E F2 B5 E8 8A 29 01 D0
F7 C8 CC F0 B3 D0 E5 68 0A A2 02 C9 50 90 04 E9 50 A2
04 69 01 0A A8 BD F1 B5 19 F2 B3 99 F2 B3 88"                F00E
200 GOSUB 520: CALL - 144                                        7C6F
210 HEX$ = "BEF0:CA 8A 29 01 D0 F0 60 B0 EC E6 44 A5 44 C9
23 90 D3 18 90 03 4C 48 BE 86 48 BD 88 C0 60 01 01 01
00 48 AD F8 06 4A 90 30 20 4D BF D0 16 AD 78 04 0A C5
2E D0 07 A9 01 99 0C BF D0 1C A9 01 8D F8": GOSUB 520
: CALL - 144                                                        A628
220 HEX$ = "BF30:06 D0 15 A9 50 8D 78 04 A9 00 20 5A BE 20
4D BF A9 00 99 0C BF 8D 78 04 A9 60 4C D4 BD A0 02 B1
48 A8 B9 0C BF 60 08 78 A9 00 85 45 AD F6 B7 85 44 4C
42 AE 20 0C FD C9": GOSUB 520: CALL - 144                        BEDA
230 HEX$ = "BF68:9B F0 01 60 68 68 68 68 4C 7F B3 A9 00 85
45 AD EA B7 85 44 4C 42 AE 00 00 00 AD 65 BF 8D 90 BF
AA A9 00 9D B8 04 20 08 C4 A2 02 BD 08 02 9D AE B3 CA
D0 F7 A9 AD 8D B1 B3 A2 02 BD 04 02 9D B1 B3 CA D0 F7
A9 AD 8D B4 B3"                                                  7265
240 GOSUB 520: CALL - 144                                        7C6F
250 HEX$ = "BFB1:4C C8 BF": GOSUB 520: CALL - 144              2AB5
260 REM ctJctJALLOCATE SECTOR DISKctJctJ

```

(suite page 26)

```

270 HEX$ = "B247:F0 2A AD F1 B5 A2 00 C9 28 90 02 A2 02 CE
    F0 B5 30 15 18 3E F3 B5 3E F2 B5 90 F2 EE EE B5 D0 03
    EE EF B5 AD F0 B5 60 A9 00 8D F1 B5 A9 00 8D 9E B3 20
    F7 AF 18 AD EB B3 6D": GOSUB 520: CALL - 144
280 HEX$ = "B280:EC B3 F0 09 CD EF B3 90": GOSUB 520: CALL
    - 144
290 HEX$ = "B288:14 A9 FF D0 0A AD 9E B3 D0 2E A9 01 8D 9E
    B3 8D EC B3 18 69 11 8D EB B3 8D F1 B5 0A A2 02 C9 50
    90 04 E9 50 A2 04 69 01 20 E0 B2 90 C6": GOSUB 520: C
    ALL - 144
300 REM ctJctJNEW VTOCctJctJ
310 HEX$ = "B2D2:20 BA BE A9 00 8D F1 B5 4C FB AF 4C BA BE
    0A A8 B9 F2 B3 9D F1 B5 F0 06 38 A9 00 99 F2 B3 88 CA
    8A 29 01 D0 EB 60": GOSUB 520: CALL - 144
320 REM ctJctJI0B TABLE + FACE BctJctJ
330 HEX$ = "BD04:A0 04 8C F8 06 A0 04 8C F8 04 A0 01 B1 48
    78 A0 0F 91 48 AA BD 89 C0 A0 02 B1 48 A8 C0 03 B0 0C
    AC 00 C4 AC 00 C5 AC 00 C6 AC 00 C7 BD 8E C0 BD 8C C0
    A0 12 BD 8C C0 48 68 48 68": GOSUB 520: CALL - 144
340 HEX$ = "BD3F:8E F8 05 DD 8C C0 D0 03 88 D0 EE 08 A0 06
    D0 05": GOSUB 520: CALL - 144

360 REM ctJctJMYSEEK PARMSctJctJ
370 HEX$ = "BE5A:0A 48 20 4D BF 6A 68 90 4C 20 6B BE 4E 78
    04 60 60": GOSUB 520: CALL - 144
380 REM ctJctJRECALIBRATE DISK HEADctJctJ
390 HEX$ = "BDD1:4C 11 BF 20 95 BE": GOSUB 520: CALL - 14
    4
400 REM PATCH ENTETE CATALOG ET MODIFIE PARMS
410 HEX$ = "ADA3:A9 16 8D 9D B3 20 2F AE 20 2F AE A2 1C BD
    E7 AE 20 C5 9F CA D0 F7 86 46 4C 8E AE": GOSUB 520: C
    ALL - 144
420 HEX$ = "AE6A:A0 2C 20 08 AF 8E C5 B5 B1 42 99 D1 B5 88
    10 F8 C8 18 60 00 A0 2C 20 08 AF B9 D1 B5 91 42 88 10
    F8 C8 38 60 20 58 BF 20 48 F9 20 73 BF A2 05 20 4A F9
    AD EF B3 85 44 20 42 AE 20 AD AE 20 AF B3 4C C9 AD A2
    06 20"
430 GOSUB 520: CALL - 144
440 HEX$ = "AEB0:4A F9 A9 F2 85 18 A9 B3 85 19 A0 A0 A9 00
    85 3C 85 3D F8 B1 18 85 42 A2 08 26 42 A9 00 65 3C 85
    3C A9 00 65 3D 85 3D CA D0 EF 88 D0 E6 D8 A5 3D 20 DA
    FD A5 3C 4C DA FD 8D C5 C5 D2 C6 AD AA AD"
450 GOSUB 520: CALL - 144
460 HEX$ = "AEF0:D3 CB C3 C1 D2 D4 AD AA AD C5 D6 C9 D2 C4
    AD AA AD CC CF D6 EA": GOSUB 520: CALL - 144
470 REM ctJctJLINE FEEDctJctJ
480 HEX$ = "B3AF:20 2F AE 20 2F AE 60": GOSUB 520: CALL -
    144
490 REM ctJctJEND PROGRAMMctJctJ
500 TEXT : PRINT "ctGctGctG": HOME : END
510 REM ctJctJSH. LAM ROUTINEctJctJ
520 HEX$ = HEX$ + " N D7D2G": FOR I = 1 TO LEN (HEX$): PO
    KE 511 + I, ASC ( MID$ (HEX$,I,1)) + 128: NEXT : POKE
    72,0: RETURN

```

DA08

B088

D5A1

1E18

D597

E412

FE7A

8053

83D7

781C

7C6F

C85D

7C6F

F8D2

59E2

DF61

BE8D

Dans son numéro 9, *TREMLIN MICRO* a proposé une courte routine en assembleur (3 lignes) permettant de détecter la frappe d'une touche du clavier. Voici un programme plus complexe permettant de tester les combinaisons de touches Pomme ouverte ou Pomme fermée. De nombreux programmes (APPLEWORKS pour ne citer que lui), utilisent l'une ou l'autre de ces touches suivie d'une lettre pour accéder aux divers menus.

# INKEY

## LES POSSIBILITÉS DE INKEY :

- Saisie au vol de tous les caractères du clavier, en minuscules ou majuscules.
- Saisie de la frappe d'une des touches pomme (ou des deux simultanées) et d'une touche alphanumérique. Dans ce cas, les minuscules sont converties en majuscules.
- La frappe d'une touche est accompagnée de l'émission d'un son variable suivant le cas.

## UTILISATION :

```
BLOAD INKEY.0
CALL 768,R
```

Suivant la (ou les) touche(s) pressée(s) la variable **R** aura la valeur :

- touche alphanumérique :  $0 \leq R \leq 127$  : code ASCII correspondant
- touche PO + touche alphanumérique :  $R = 256 +$  code ASCII
- touche PF + touche alphanumérique :  $R = 512 +$  code ASCII
- touche PF + touche PO + touche alphanumérique :  $R = 768 +$  code ASCII.

Dans tous les cas les caractères de contrôle sont acceptés. Le programme BASIC DEMO.INKEY visualise parfaitement les différentes possibilités.

## COMMENTAIRES SUR LE PROGRAMME :

INKEY utilise des routines de la ROM de l'APPLE qui méritent quelques explications :

**PTRGET (\$DFE3)** : Recherche l'adresse d'une variable. Si cette variable n'a pas encore été définie, elle est créée en mémoire et son adresse se trouve dans **VARPNT** (\$83-84). Pour voir si une touche Pomme ou un bouton de manette a été pressé, lire en **\$C061** (PO ou bouton 0) ou en **\$C062** (PF ou bouton 1).

### Conversion d'une minuscule en majuscule

Codage de A en ASCII : \$41 = 01 000001

Codage de a en ASCII : \$61 = 01 100001

Ces deux octets ne diffèrent que par le bit 5. Il suffit donc de faire un **AND \$DFD** (11011111) pour convertir une minuscule en majuscule.

Récupération de la valeur finale dans la variable **R** : on charge maintenant **Y** avec l'octet de poids faible et **A** avec l'octet de poids fort. Un appel de **GIVAYF** (\$E2F2) transfère cette valeur dans **FAC** (accumulateur flottant). La routine **MOVFM** effectue le transfert du **FAC** vers la variable **R** si l'on a eu soin de charger dans **X** le contenu de **VARPNT** et dans **Y** celui de **VARPNT + 1** (vous n'avez pas oublié que l'adresse de la variable **R** se trouve dans **VARPNT** depuis l'appel de **PTRGET**).

### Quelques applications de INKEY

1. Saisie au vol de caractères frappés au clavier en évitant la saturation de la mémoire de variables alphanumériques, ce qui ne manque pas d'arriver si l'on utilise trop souvent la fonction **GET R\$** de l'Applesoft. Mais pour cette seule utilisation on peut faire plus simple.
2. Saisie de commandes avec les touches Pomme. Dans ce cas la conversion des minuscules en majuscules est indispensable. En effet votre programme se plante lorsque vous frappez une commande avec la touche Caps Lock non enfoncée, si vous n'avez pas prévu le traitement d'une commande en minuscules.
3. Avec un programme d'écriture de caractères en haute résolution, on peut envisager l'utilisation aisée de plusieurs polices, par exemple les caractères ASCII standard obtenus par frappe de la touche correspondante, les caractères grecs obtenus par PO + touche, les caractères en indice par PF + touche, etc.  
... et d'autres utilisations que vous ne manquerez pas de trouver !

(suite page 28)

## DEMO.INKEY

```

100 REM DEMO.INKEY - R.JOST 1987
110 TEXT : NORMAL : HOME
120 PRINT CHR$(4)"BLOAD "INKEY.0"
130 TEXT : PRINT "Essayez les différentes touches ...."
140 CALL 768,R
150 IF R = 27 THEN 210
160 IF R < 128 THEN PRINT R. CHR$(R): GOTO 140
170 IF R < 512 THEN PRINT "POMME OUVERTE - " CHR$(R - 256): GOTO 140
180 IF R < 768 THEN PRINT "POMME FERMEE " - " CHR$(R - 512): GOTO 140
190 PRINT " LES 2 POMMES ENFONCEES - " CHR$(R - 768)
200 GOTO 140
210 HOME : VTAB 22: PRINT "<M>ENU DE DISQUETTE c<tG";: GET R$: PRINT : IF R
  $ = "M" OR "R" = "m" THEN PRINT CHR$(4)"RUN /TME/MENU"

```

7F31  
2EEA  
2781  
44AF  
3C8F  
7B6B  
36C9  
217B  
0E60  
2D40  
B619

### EDASM ProdOS

```

1 * SAISIE D'UNE TOUCHE
2 *
3 CODE EQU $00 : octet haut du code
4 DUREE EQU $06 : durée du son
5 HAUTEUR EQU $07 : hauteur (fréquence) du son
6 VARPNT EQU $83 : adresse de la dernière variable
7 *
8 KBD EQU $C000 : lit le clavier
9 STROBE EQU $C010 : réinitialise lect. du clavier
10 SPKR EQU $C030 : active le haut-parleur.
11 BUTTON0 EQU $C061 : lecture état touche pomme ouv.
12 BUTTON1 EQU $C062 : lecture état touche pomme fer.
13 CHKCOM EQU $DEBE : teste la virgule
14 PTRGET EQU $DFE3 : rech. d'une var. par son nom
15 GIVAYF EQU $E2F2 : rend flottant l'entier en A,Y.
16 COMBYTE EQU $E74C : teste la virgule, évalue la
17 : variable et stocke dans X.
18 MOVMF EQU $EB2B : transfère FAC dans octet
19 : pointé par X,Y
20 ORG $300
21 *
22 * KEY
23 *
24 * Appel par CALL KEY,R : le code ASCII de la touche
25 * se trouve dans R, augmenté de :
26 * - 256 si POMME OUVERTE a été pressée en meme temps
27 * - 512 si POMME FERMEE a été pressée
28 * - 768 si les deux touches POMME sont sollicitées
29 *
30 *
31 JSR CHKCOM : virgule présente?
32 JSR PTRGET : oui, recherche l'adresse de R
33 LDA £$20 : initialise
34 STA DUREE : les paramètres
35 STA HAUTEUR : de la routine SON.
36 LDA £00 : met à zéro le drapeau testant
37 STA CODE : l'appui d'une touche POMME.
38 KEY LDA KBD : a.t.on enfoncé une touche?

```

Si vous ne possédez pas d'assembleur, contentez-vous de taper les codes ci-dessous.

300: 20 BE DE  
303: 20 E3 DF  
306: A9 20  
308: 85 06  
30A: 85 07  
30C: A9 00  
30E: 85 00  
310: AD 00 C0

313:	10 FB	39	BPL	KEY	: non
315:	2C 10 C0	40	BIT	STROBE	: oui , reset du clavier
318:	48	41	PHA		: sauve A temporairement
319:	AD 61 C0	42	LDA	BUTTON0	: touche POMME OUVERTE
31C:	10 0E	43	BPL	POMFERM	: non sollicitée
31E:	68	44	PLA		: rappelle A
31F:	C9 E0	45	CMP	£\$E0	: est-ce une minuscule
321:	90 02	46	BCC	B0	: non
323:	29 DF	47	AND	£\$DF	: oui: on la convertit en majusc.
325:	48	48	PHA		: sauve A
326:	E6 00	49	INC	CODE	: touche POMME OUVERTE utilisée.
328:	A9 40	50	LDA	£\$40	: le son sera en conséquence..
32A:	85 07	51	STA	HAUTEUR	
32C:	AD 62 C0	52	POMFERM LDA	BUTTON1	: la touche POMME FERMEE
32F:	10 10	53	BPL	SORTIE	: na pas été utilisée
331:	68	54	PLA		: si , récupère A
332:	C9 E0	55	CMP	£\$E0	: est-ce une minuscule?
334:	90 02	56	BCC	B1	: non
336:	29 DF	57	AND	£\$DF	: oui: on la convertit en majusc.
338:	48	58	PHA		: et on sauve dans la pile.
339:	E6 00	59	INC	CODE	: rajoute 512
33B:	E6 00	60	INC	CODE	: au code
33D:	A9 60	61	LDA	£\$60	: le son sera différent..
33F:	85 07	62	STA	HAUTEUR	
341:	20 5F 03	63	SORTIE JSR	SONX	: appel routine SON
344:	68	64	PLA		
345:	29 7F	65	AND	£\$7F	: met bit 7 à zéro
347:	A8	66	TAY		: transfère A dans Y
348:	A5 00	67	LDA	CODE	: récupère l'octet de poids fort
34A:	20 F2 E2	68	JSR	£E2F2	: transfère dans FAC
34D:	A6 83	69	LDX	VARPNT	: adresse de la variable R
34F:	A4 84	70	LDY	VARPNT+1	
351:	20 2B EB	71	JSR	MOVMF	: transfère FAC dans R.
354:	60	72	RTS		: c'est termine é
		73	*		
		74	* SON		
		75	*		
		76	* Appel par CALL SON,durée,hauteur		
		77	*		
355:	20 4C E7	78	SON JSR	COMBYTE	: attend une virgule
358:	86 06	79	STX	DUREE	: et stocke la variable.
35A:	20 4C E7	80	JSR	COMBYTE	: teste la virgule, évalue
35D:	86 07	81	STX	HAUTEUR	: et stocke la variable
35F:	A4 06	82	SONX LDY	DUREE	: début du programme SON
361:	AD 30 C0	83	SON00 LDA	SPKR	: active le haut-parleur
364:	88	84	SON0 DEY		: tant que
365:	D0 04	85	BNE	SON1	: Y > 0
367:	C6 06	86	DEC	DUREE	
369:	F0 07	87	BEQ	FINSON	
36B:	CA	88	SON1 DEX		
36C:	D0 F6	89	BNE	SON0	
36E:	A6 07	90	LDX	HAUTEUR	
370:	D0 EF	91	BNE	SON00	
372:	60	92	FINSON RTS		

**BSAVE INKEY.0,AS300,LS73**

# LES CARRÉS MAGIQUES

## HISTORIQUE

- Les CARRÉS MAGIQUES sont fort anciens. Ils étaient connus des Chinois et des Hindous bien avant notre ère, sans que l'on puisse en fixer l'origine exacte.
- Le carré d'ordre 3 existe dans un manuscrit ARABE du 8<sup>e</sup> siècle, mais c'est au 14<sup>e</sup> siècle qu'il fut amené en Europe par le grammairien Byzantin MOSCHOPoulos.
- PASCAL, FERMAT, EULER, FRENICLE, VIOLLE et bien d'autres, ont étudié mathématiquement le problème et donnent des constructions plus ou moins simples, mais on n'a pas encore découvert de méthode générale pour les construire tous.

Le programme CARRES.MAG généralise le problème et permet de construire n'importe quel carré magique, MAIS j'ai limité à 19 l'ordre d'un carré à cause de la grandeur de l'écran.

On pourrait modifier le programme pour construire des carrés d'ordre supérieur.

Pour obtenir des CARRÉS MAGIQUES, quel que soit l'ordre, j'ai mis au point trois méthodes :

1. Carrés IMPAIRS : méthode dérivée de celle de LA LOUBERE (XVII<sup>e</sup> siècle), puis de celle de BACHET ;
2. Carrés PAIRS multiples de 4 : méthode inspirée de la précédente ;
3. Carrés PAIRS non multiples de 4 : méthode des pourtours qui utilise des carrés magiques multiples de 4, que l'on entoure d'une enceinte calculée selon un procédé assez complexe.

**BIBLIOGRAPHIE SUCCINCTE** W. ROUSE-BALL Récréations Mathématiques (PARIS 1926/27) — Maurice KRAITCHIK Traité des carrés magiques (GAUTHIER—VILLARS 1930) — Eutrope CAZALAS Carrés magiques au degré n (HERMANN 1934) — Philip. de LAHIRE Nouvelles constructions et considérations sur les carrés magiques (PARIS 1706) — FRENICLE Des carrés ou tables magiques (PARIS 1729) — Edouard LUCAS Les carrés magiques de FERMAT et de FRENICLE (PARIS DELAGRAVE 1887). ■

## PROPRIÉTÉS

- Un carré magique reste magique :
  1. Si on fait tourner tous ses nombres d'un angle DROIT par rapport au centre ;
  2. Si on pratique une symétrie par rapport à la médiane verticale ou à la médiane horizontale, ou par rapport à une diagonale ;
  3. Par échange de 2 lignes et de 2 colonnes équidistantes du centre ;
  4. Par symétrie par rapport au centre.
- De nombreux mathématiciens ou chercheurs étudièrent le problème et ils introduisirent des conditions supplémentaires. C'est ainsi qu'ils découvrirent :
  - LES CARRÉS BIMAGIQUES qui restent magiques en remplaçant leurs nombres par leurs carrés ;
  - LES CARRÉS TRIMAGIQUES qui restent magiques en remplaçant leurs nombres par leurs carrés ou par leurs cubes.

## GÉNÉRALITÉS

- Si N est l'ordre (nombre de colonnes ou de lignes), le carré contient les  $N \times N$  nombres entiers, de telle façon que les sommes des lignes, des colonnes et des diagonales représentent le même nombre ou **CONSTANTE MAGIQUE**.
- Soit S la somme des N premiers nombres :  

$$S = 1 + 2 + 3 + \dots + (N \times N)$$

$$S = (N \times N) + [(N - 1) \times (N - 1)] + \dots + 2 + 1$$

$$2 \times S = (N \times N) \times (N \times N + 1) \text{ d'où}$$

$$S = (N \times N) \times (1 + N \times N) / 2$$
- La **CONSTANTE MAGIQUE** est donc :  

$$CM = N \times (1 + N \times N) / 2$$
- Les constantes des premiers CARRÉS MAGIQUES sont donc :  
 pour  $N=3 \rightarrow CM=15$     $N=4 \rightarrow CM=34$     $N=5 \rightarrow CM=65$   
 pour  $N=6 \rightarrow CM=111$     $N=7 \rightarrow CM=175$  etc.

## GÉNÉRALISATION à des nombres non consécutifs.

Soit **OD** l'ordre, **NB** le plus petit nombre et **INC** l'incrément (différence entre 2 nombres consécutifs de la suite). On obtient :  

$$S = (OD \times OD) \times (2 \times NB + (OD \times OD + 1) \times INC) / 2$$

$$CM = OD \times (2 \times NB + (OD \times OD + 1) \times INC) / 2$$

```

100 REM ----- CARRE MAGIQUE -----
105 : 003A
110 PRINT CHR$(4)"PR£3": PRINT ED56
115 PRINT " ORDRE ...= Nombre de lignes du carré magique
e...limité ici à "; INVERSE : PRINT " 19 ": NORMAL 3C87
120 PRINT " INCREMENT = Différence constante entre 2 nom
bres de la suite.": PRINT E458
125 PRINT : PRINT "Lorsque tous les nombres du CARRE MAG
IQUE ne pourront pas tenir dans l'écran": PRINT "APP
LE refusera les données, et vous devrez entrer des n
ombres plus petits." 4627
130 VTAB 13: INPUT "-Entrez le plus petit nombre :";NB:
PRINT FEE4
135 INPUT "-Entrez l'incrément :";INC: PRINT C2BE
140 INPUT "-Quel est l'ordre du CARRE ? ";OD 81DA
145 IF OD = 2 OR OD > 19 THEN CALL - 998: CALL - 198:
GOTO 140 9E4A
150 ND = NB + (OD ^ 2 - 1) * INC:NN = LEN ( STR$(ND)) 7EDB
155 DIM A(OD + 1,OD + 1),H(OD),V(OD) 351A
160 IF (NN + 1) * OD > 77 THEN GOTO 265 0A39
165 IF OD / 2 < > INT (OD / 2) THEN GOSUB 285: GOTO 3
50: REM ORDRE IMPAIR C621
170 IF OD / 4 < > INT (OD / 4) THEN GOTO 295: REM NON
MULTIPLES DE 4 71A4
175 : 003A
180 REM ----- CARRES PAIRS MULTIPLES DE 4 -----
185 : 003A
190 GOSUB 285 234F
195 K = 1: IF AA = 1 THEN K = 2 * N - 1 08D9
200 FOR J = 1 TO OD: FOR I = 1 TO OD 9279
205 A1 = NB + (K - 1) * INC:A(J,I) = A1:L1 = LEN ( STR$(
A1)) 3E03
210 K = K + 1 215F
215 IF L1 < = L THEN 225 7774
220 L = L1 7999
225 NEXT I,J 9741
230 FOR J = 1 TO OD / 2: FOR I = 1 TO OD 3376
235 IF (J = 1 OR J = 4 OR J = 5 OR J = 8 OR J = 9) AND (
I = 1 OR I = 4 OR I = 5 OR I = 8 OR I = 9 OR I = 12
OR I = 13 OR I = 16) THEN 250 C1DF
240 IF (J = 2 OR J = 3 OR J = 6 OR J = 7 OR J = 10) AND
(I = 2 OR I = 3 OR I = 6 OR I = 7 OR I = 10 OR I = 1
1 OR I = 14 OR I = 15) THEN 250 772E
245 Z = A(OD - J + 1,OD - I + 1):A(OD - J + 1,OD - I + 1
) = A(J,I):A(J,I) = Z DE30
250 NEXT I,J 9741
255 IF AA = 1 THEN 315 038D
260 GOTO 415 0F45
265 HOME : CALL - 198: VTAB 8: PRINT "Plus petit nombre
= ";NB;; PRINT " ""Incrément = ";INC;; PRINT " ""
"Ordre = ";OD 3663
270 PRINT : PRINT "L'une de ces données (au moins) est t
rop grande...Recommencez..." 3A8E

```



```

275 VTAB 21: POKE 1403,50: PRINT "Une touche...SUP ";: C
  ALL - 198: GET H#: PRINT                                A2C0
280 PRINT : HOME : RUN 115                                2A08
285 HOME : VTAB 10: POKE 1403,28: PRINT "PATIENCE... Je
  calcule...": RETURN                                    A86D
290 :                                                      003A
295 REM ----- CARRES PAIRS NON MULTIPLES DE 4 -----
300 :                                                      003A
305 DIM B(OD + 1,OD + 1)                                  7E5D
310 N = OD:OD = OD - 2:AA = 1:: GOTO 190                 8D52
315 OD = N                                                83B1
320 FOR J = 1 TO OD: FOR I = 1 TO OD                     9279
325 B(J + 1,I + 1) = A(J,I)                              B665
330 A(J,I) = B(J,I)                                       8C73
335 NEXT : NEXT                                          143E
340 GOTO 560                                              F046
345 :                                                      003A
350 REM ----- CARRES IMPAIRS -----
355 :                                                      003A
360 X = INT (OD / 2) + 1:Y = 1: REM PLACE DE NB DANS LE
  CARRE                                                  6969
365 FOR I = NB TO ND STEP INC                             D81E
370 A(X,Y) = I:L = LEN ( STR# (I))                       7790
375 ZX = X:ZY = Y:X = X - 1:Y = Y - 1                   A25A
380 IF X < 1 THEN X = OD                                  4286
385 IF Y < 1 THEN Y = OD                                  8188
390 IF A(X,Y) < > 0 THEN X = ZX:Y = ZY + 1              5C99
395 NEXT                                                  0582
400 :                                                      003A
405 REM ----- ECRITURE SUR ECRAN -----
410 :                                                      003A
415 HOME : PRINT :D1 = 0:D2 = 0                          81EA
420 LL = 1: IF OD < 14 THEN LL = 2                      61A7
425 FOR I = 1 TO OD                                       171F
430 H(I) = 0:V(I) = 0                                     3E0C
435 FOR J = 1 TO OD                                       7720
440 A(J,I) = INT (A(J,I))                                CF96
445 VTAB I: POKE 1403,J * (L + LL) - LEN ( STR# (A(J,I)
  ))                                                    E466
450 PRINT A(J,I)                                         C30B
455 H(I) = H(I) + A(J,I)                                 65AD
460 V(I) = V(I) + A(I,J)                                 A6C9
465 NEXT J                                               A3CC
470 D1 = D1 + A(I,I)                                     B8D2
475 D2 = D2 + A(I,OD - I + 1)                           F529
480 NEXT I                                               9DCB
485 Z = OD * (2 * NB + ((OD ^ 2) - 1) * INC) / 2:Z = IN
  T (Z)                                                  CEDC
490 PRINT : PRINT "NOMB.INFERIEUR:";: INVERSE : PRINT "
  "NB" ";: NORMAL                                       51A0
495 PRINT " "INCREMENT:";: INVERSE : PRINT " "INC" ";: N
  ORMAL                                                 4DD8
500 PRINT " "ORDRE:";: INVERSE : PRINT " "OD" ";: NORMAL 1968

```

505 PRINT " "CONST.MAGIQ.:";: INVERSE : PRINT " "Z" " ;:	AF98
NORMAL : PRINT " " ;	9F42
510 FOR I = 1 TO 40:Z = PEEK (49200): NEXT	
515 PRINT : PRINT : PRINT "VOULEZ-VOUS VOIR D'AUTRES CAR	
RES MAGIQUES ? (O/N) " ;: GET H\$: PRINT : PRINT	58A3
520 IF H\$ = "N" THEN PRINT "Confirmez en appuyant sur "	
;: INVERSE : PRINT " N " ;: NORMAL : PRINT " " ;	B987
525 IF H\$ = "O" THEN PRINT "Confirmez en appuyant sur "	
;: INVERSE : PRINT " O " ;: NORMAL : PRINT " " ;	DE09
530 FOR I = 1 TO 20:Z = PEEK (49200): NEXT	E040
535 GET H\$: PRINT	1B1E
540 IF H\$ = "O" THEN HOME : RUN 115	0754
545 PRINT CHR\$(4)"RUN MENU"	B1B4
550 END	0180
555 :	003A
560 REM ----- CALCUL DES NOMBRES DU POURTOUR -----	
565 :	003A
570 FOR I = 1 TO N - 3	34D6
575 A(I + 1,1) = NB + 2 * I * INC:A(I + 1,N) = NB + (N ^	
2 - 2 * I - 1) * INC	3F3B
580 A(1,N - I) = NB + (2 * I + 1) * INC:A(N,N - I) = NB	
+ (N ^ 2 - 2 * I - 2) * INC	72C2
585 IF I > N / 2 - 2 THEN A(1,N - I) = NB + (2 * I + 3)	
* INC:A(N,N - I) = NB + (N ^ 2 - 2 * I - 4) * INC	8095
590 IF INT (I / 2) < > I / 2 THEN Z = A(I + 1,1):A(I +	
1,1) = A(I + 1,N):A(I + 1,N) = Z	4957
595 IF I = N - 3 THEN 605	7C6F
600 IF INT (I / 2) < > I / 2 THEN Z = A(1,N - I):A(1,N	
- I) = A(N,N - I):A(N,N - I) = Z	24CF
605 NEXT	0582
610 A(1,1) = NB:A(N,N) = NB + (N ^ 2 - 1) * INC	0977
615 A(N - 1,N) = NB + INC:A(N - 1,1) = NB + (N ^ 2 - 2)	
* INC	592B
620 A(N,1) = NB + (N - 1) * INC:A(1,N) = NB + (N ^ 2 - N	
) * INC	2999
625 A(N,2) = NB + 2 * (N - 2) * INC:A(1,2) = NB + (N ^ 2	
- 2 * N + 3) * INC	0272
630 AA = 0	DE82
635 GOTO 415	0F45

### Les meilleures routines LM ne sont pas les plus longues.

C'est la raison pour laquelle nous reprenons périodiquement, dans nos ouvrages collectifs, la plupart des programmes courts insérés dans *Tremplin Micro*. Mais ces ouvrages contiennent aussi de nouvelles versions de routines LM déjà publiées. C'est une raison suffisante pour ne pas manquer :

## **NOUVELLES ROUTINES POUR LE 65C02 DE L'APPLE**

(on sait que le 65C816 du GS émule parfaitement le 65C02)

Prix de vente avec disquette : 160 F

# FOND D'ÉCRAN (65C02)

## QUESTION :

Comment remplacer le petit programme Basic ci-contre par une routine en langage machine... pour une plus grande rapidité d'exécution ?

```
10 HOME : A$ = ""
20 FOR A = 1 TO 40: A$ = A$ + "WV": NEXT
30 FOR B = 23 TO 1 STEP - 1: VTAB B: PRINT
  CHR$(27):: INVERSE : PRINT A$: NEXT
40 NORMAL : PRINT CHR$(24)
```

## RÉPONSE :

Voici une solution en 33 ou 27 octets. Avec un écran couleur, elle donne un fond d'écran plutôt joli. On peut aussi essayer avec la valeur \$4E en \$311 (FEC0) ou \$309 et 0310 (FEC1). La partie DÉMO ci-dessous est tout à fait superflue, comme on le constatera à l'usage.

```
100 PRINT CHR$(4)"PRÉ3": PRINT : GOSUB 220
110 REM GOSUB 240 POUR ROUTINE $F847 (GBASCALC)
120 VTAB 12: PRINT "FOND D'ECRAN EN 80 COLONNES " : GOSUB 200:
  CALL 768
130 GOSUB 210: HOME
140 PRINT CHR$(17): VTAB 12: PRINT "FOND D'ECRAN EN 40 COLON
  NES": PRINT : GOSUB 200: CALL 768
150 GOSUB 210
160 VTAB 21: CALL - 958: PRINT : PRINT "(E)NCORE (M)ENU DISK (F)IN " :
  GET R$: PRINT R$
170 HOME : IF R$ = "E" OR R$ = "e" THEN PRINT CHR$(18): GOTO 120
180 IF R$ = "M" OR R$ = "m" THEN PRINT CHR$(4)"RUN MENU"
190 END
200 PRINT "(PRESSER UNE TOUCHE)"
210 CALL - 198: POKE 49168,0: WAIT 49152,128: POKE 49168,0: PRINT :
  RETURN
220 FOR I = 768 TO 800: READ R : POKE I,R: NEXT : RETURN
230 DATA 169,23,133,37,32,193,251,160,39,169,87,145,40,44,85,192,169,
  86,145,40,44,84,192,136,16,239,198,37,165,37,16,226,96
240 REM ROUTINE UTILISANT GBASCALC (4 OCTETS DE MOINS)
250 FOR I = 768 TO 796: READ R: POKE I,R: NEXT : RETURN
260 DATA 162,23,138,32,71,248,160,39,169,87,145,38,44,85,192,169,
  86,145,38,44,84,192,136,16,239,202,16,230,96
```

### Avec BASCALC

```
300 : A9 17      LDA £$17
302 : 85 25      STA $25
304 : 20 C1 FB   JSR $FBC1
307 : A0 27      LDY £$27
309 : A9 57      LDA £$57
30B : 91 28      STA ($28),Y
30D : 2C 55 C0   BIT $C055
310 : A9 56      LDA £$56
312 : 91 28      STA ($28),Y
314 : 2C 54 C0   BIT $C054
317 : 88         DEY
318 : 10 EF      BPL $0309
31A : C6 25      DEC $25
31C : A5 25      LDA $25
31E : 10 E2      BPL $0302
320 : 60         RTS
```

B\$AVE FEC0,A\$300,LS21

### Avec GBASCALC

```
300 : A2 17      LDX £$17
302 : 8A         TXA
303 : 20 47 F8   JSR $F847
306 : A0 27      LDY £$27
308 : A9 57      LDA £$57
30A : 91 26      STA ($26),Y
30C : 2C 55 C0   BIT $C055
30F : A9 56      LDA £$56
311 : 91 26      STA ($26),Y
313 : 2C 54 C0   BIT $C054
316 : 88         DEY
317 : 10 EF      BPL $0308
319 : CA         DEX
31A : 10 E6      BPL $0302
31C : 60         RTS
```

B\$AVE FEC1,A\$300,LS1D

APPLE II GS  
exclusivement



# NON-ACCÈS

## au tableau de bord

Voici, en Basic, un exemple de programmation montrant comment changer le standard de traitement de l'interruption générée par CTRL-PO-ESC (accès au tableau de bord de l'Apple IIGS). Les explications sont résumées dans le listage du programme.

### TABLORD

```

100 TEXT : NORMAL :D$ = CHR$(4): PRINT D$"PRÉ3"                5506
105 REM *** LE TABLEAU DE BORD DU GS ***
110 :                                                            003A
115 PRINT " "IL EST POSSIBLE DE MODIFIER 'au vol' LA CONFIGURATION DU
    GS"                                                            92A5
120 PRINT "GRACE A L'ACCESSOIRE DE BUREAU: Control Panel"       99CD
125 PRINT : PRINT "ESSAYEZ "*****"en appuyant sur CTRL-PO-ESC"  9E3E
130 PRINT " "*****"puis en choisissant Control Panel dans le Menu"  C877
135 PRINT                                                         75BA
140 PRINT                                                         75BA
145 PRINT "Mais, SI on NE veut PAS de cette liberté ?"         53E2
150 PRINT " "*****"ALORS il faut modifier le vecteur IRQ.DSK.ACC"  87B3
155 PRINT " "*****"et par exemple faire exécuter CLC RTL au lieu d'un sa
    ut"                                                            4260
160 PRINT " "*****"à la routine standard qui gère les accessoires de bur
    eau"                                                            57D1
165 PRINT : PRINT "TAPEZ SUR Return POUR NEUTRALISER LES ACCESSOIRES D
    E BUREAU ";                                                    27A4
170 INPUT " ";X$                                                 AE7F
175 :                                                            003A
180 REM En STANDARD, IRQ.DSK.ACC contient JMP FEADF2
185 REM PAS D'ACCES: IRQ.DSK.ACC ne fera s'exécuter que CLC RTL
190 REM SOUS-PROGRAMME POUR MODIFIER IRQ.DSK.ACC
195 :                                                            003A
200 READ I: POKE 768 + J,I:J = J + 1: IF I = 107 THEN 285       8F81
205 GOTO 200                                                       333D
210 :                                                            003A
215 REM IL EST IMPLANTE A PARTIR DE $00/0300
220 :                                                            003A
225 DATA 24,251,194,48,244,0,0,244,0,0,244,18,0,162,3,17,34,0,0,225,10
    4,141,58,3,104,141,55,3,244,18,0,244,0,0,244,70,3,162,3,16,34,0,0,
    225,56,251,96                                                F627
230 :                                                            003A
235 REM RETOUR A LA NORMALE EN $00/032F
240 :                                                            003A

```

```

245 DATA 24,251,194,48,244,18,0,244,0,0,244,47,3,162,3,16,34,0,0,225,5 93BD
      6,251,96 003A
250 :
255 REM NOUVEAU S/P APPELE PAR IRQ.DSK.ACCESS EN $00/0346
260 : 003A
265 DATA 24,107 EDAD
270 : 003A
275 REM PAS D'ACCES AUX ACCESSOIRES
280 : 003A
285 CALL 768 8331
290 PRINT : PRINT "POUR LES RETROUVER,TAPER CALL 815 et CTRL-PO-ESC" B1AE

```

```

*0=m:0=x
*00/300L
0=m 0=x 1=LCbank (0/1)

```

Pour obtenir les listages ci-après, tapez d'abord  
0=m : 0=x.

```

00/0300: 18 CLC
00/0301: FB XCE
00/0302: C2 30 REP £30
00/0304: F4 00 00 PEA 0000
00/0307: F4 00 00 PEA 0000
00/030A: F4 12 00 PEA 0012
00/030D: A2 03 11 LDX £1103
00/0310: 22 00 00 E1 JSL E10000
00/0314: 68 PLA
00/0315: 8D 3A 03 STA 033A
00/0318: 68 PLA
00/0319: 8D 37 03 STA 0337
00/031C: F4 12 00 PEA 0012
00/031F: F4 00 00 PEA 0000
00/0322: F4 46 03 PEA 0346
00/0325: A2 03 10 LDX £1003
00/0328: 22 00 00 E1 JSL E10000
00/032C: 38 SEC
00/032D: FB XCE
00/032E: 60 RTS

```

Passage en mode natif  
avec des registres sur 16 bits.

Sauvegarde de l'ancien vecteur :

GetVector N=\$0012

Le résultat modifiera le champ  
opérande des instructions \$336 et \$339.

Mise en place du nouveau vecteur :

SetVector N=\$0012 P=\$000346

Passage en mode émulation  
avant le retour au Basic.

```

*L
0=m 0=x 1=LCbank (0/1)

```

```

00/032F: 18 CLC
00/0330: FB XCE
00/0331: C2 30 REP £30
00/0333: F4 12 00 PEA 0012
00/0336: F4 FE 00 PEA 00FE
00/0339: F4 F2 AD PEA ADF2
00/033C: A2 03 10 LDX £1003
00/033F: 22 00 00 E1 JSL E10000
00/0343: 38 SEC
00/0344: FB XCE
00/0345: 60 RTS
00/0346: 18 CLC
00/0347: 6B RTL

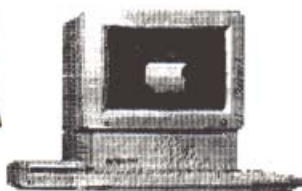
```

Restauration de l'ancien vecteur :

SetVector N=\$0012 P résultant de GetVector.

Nouveau Vecteur : Pas de réaction souhaitée.  
C=0 informe que l'interruption a été traitée.

APPLE II GS  
exclusivement



# TIC-TAC

Ce programme ressemble beaucoup au précédent... et pour cause ! Il vous promet un tic-tac assez impressionnant, scandant le fonctionnement du GS ! Notez qu'après l'avoir interrompu par POKE 756,7 et CALL 751, vous pourrez le reprendre par POKE 756,6 et CALL 751... si vous n'avez pas écrasé votre routine entre temps, bien sûr !

```

100 TEXT : NORMAL :D$ = CHR$(4): PRINT D$"PRÉ3"                55D6
105 REM *** UN TIC-TAC POUR LE GS ***
110 :                                                            003A
115 PRINT " "A PARTIR DE L'INTERRUPTION 'Timer 1sec'"          817B
120 PRINT "GRACE AUX FONCTIONS GetVector, SetVector et IntSource" 8DB4
125 PRINT " "- GetVector pour sauvegarder l'ancienne valeur de IRQ.1SEC
    C"                                                            49AF
130 PRINT " "- SetVector pour fixer un nouveau S/P qui se déclenchera" 8671
135 PRINT " ""chaque seconde, mais à condition que l'interruption ait
    été"                                                            E9AC
140 PRINT " ""autorisée par IntSource N°0006"                   D5D1
145 :                                                            003A
150 REM EN STANDARD, IRQ.1SEC contient JMP FFB498 (SEC RTL)
155 REM SOUS-PROGRAMME DU MEME MODELE QUE LE PRECEDENT
160 :                                                            003A
165 READ I: POKE 768 + J,I:J = J + 1: IF I = 107 THEN 235      817C
170 GOTO 165                                                    4347
175 :                                                            003A
180 REM IL EST IMPLANTE A PARTIR DE $00/0300
185 :                                                            003A
190 DATA 24,251,194,48,244,0,0,244,0,0,244,21,0,162,3,17,34,0,0,225,10
    4,141,58,3,104,141,55,3,244,21,0,244,0,0,244,70,3,162,3,16,34,0,0,
    225,56,251,96                                                561B
195 :                                                            003A
200 REM RETOUR A LA NORMALE EN $00/032F
205 :                                                            003A
210 DATA 24,251,194,48,244,21,0,244,0,0,244,47,3,162,3,16,34,0,0,225,5
    6,251,96                                                       A0B7
215 :                                                            003A
220 REM NOUVEAU S/P APPELE PAR IRQ.1SEC EN $00/0346
225 :                                                            003A
230 DATA 226,48,160,96,162,128,202,208,253,44,48,192,136,208,245,169,6
    4,28,50,192,24,107                                             EBD9
235 CALL 768                                                    8331
240 :                                                            003A
245 REM AUTORISATION DE L'INTERRUPTION EN $00/0002EF          (suite page 38)

```

```

250 :                                003A
255 READ I: POKE 751 + K,I:K = K + 1: IF K > 16 THEN 275 644B
260 GOTO 255                            1B47
265 :                                003A
270 DATA 24,251,194,48,244,6,0,162,3,35,34,0,0,225,56,251,96 4C51
275 CALL 751                            7F29
280 :                                003A
285 PRINT : PRINT "POUR ARRETER LE TIC-TAC, TAPEZ:         FF0B
290 PRINT " "- POKE 756,7 puis CALL 751"                0C39
295 PRINT : PRINT "POUR ANNULER L'OPERATION, TAPEZ:"    5EBB
300 PRINT " "- CALL 815"                                C865

```

\*0=m:0=x

\*00/2EFL

0=m 0=x 1=LCbank (0/1)

```

00/02EF: 18          CLC
00/02F0: FB          XCE
00/02F1: C2 30      REP £30
00/02F3: F4 06 00   PEA 0006
00/02F6: A2 03 23   LDX £2303
00/02F9: 22 00 00 E1 JSL E10000
00/02FD: 38          SEC
00/02FE: FB          XCE
00/02FF: 60          RTS
00/0300: 18          CLC
00/0301: FB          XCE
00/0302: C2 30      REP £30
00/0304: F4 00 00   PEA 0000
00/0307: F4 00 00   PEA 0000
00/030A: F4 15 00   PEA 0015
00/030D: A2 03 11   LDX £1103
00/0310: 22 00 00 E1 JSL E10000
00/0314: 68          PLA
00/0315: 8D 3A 03   STA 033A
00/0318: 68          PLA

```

#### Autorisation d'interruption à IRQ.1SEC

IntSource N=\$0006

#### Sauvegarde de l'ancien vecteur :

GetVector N=\$0015

\*L

0=m 0=x 1=LCbank (0/1)

```

00/0319: 8D 37 03   STA 0337
00/031C: F4 15 00   PEA 0015
00/031F: F4 00 00   PEA 0000
00/0322: F4 46 03   PEA 0346
00/0325: A2 03 10   LDX £1003
00/0328: 22 00 00 E1 JSL E10000
00/032C: 38          SEC
00/032D: FB          XCE
00/032E: 60          RTS
00/032F: 18          CLC
00/0330: FB          XCE
00/0331: C2 30      REP £30
00/0333: F4 15 00   PEA 0015

```

#### Mise en place du nouveau vecteur :

SetVector N=\$0015

#### Restauration :

SetVector N=\$0015

00/0336:	F4 00 00	PEA 0000
00/0339:	F4 46 03	PEA 0346
00/033C:	A2 03 10	LDX £1003
00/033F:	22 00 00 E1	JSL E10000
00/0343:	38	SEC
00/0344:	FB	XCE
00/0345:	60	RTS

\*1=m: 1=x  
 \*346L  
 1=m 1=x 1=LCbank (0/1)

00/0346:	E2 30	SEP £30
00/0348:	A0 60	LDY £60
00/034A:	A2 80	LDX £80
00/034C:	CA	DEX
00/034D:	D0 FD	BNE 034C é-03è
00/034F:	2C 30 C0	BIT C030
00/0352:	88	DEY
00/0353:	D0 F5	BNE 034A é-0Bè
00/0355:	A9 40	LDA £40
00/0357:	1C 32 C0	TRB C032
00/035A:	18	CLC
00/035B:	6B	RTL

### Restauration :

(les champs opérands des PEA sont modifiés lors de la sauvegarde de l'ancien vecteur).

SetVector N=\$0015

1 = m : 1 = x (listage normal)

### Nouveau S/P déclenché chaque seconde

TIC-TAC

Remise à 0 de l'interruption.

Référence bibliographique : CLEFS POUR APPLE II GS (2<sup>e</sup> édition) — PSI.

## STAGE INTERNATIONAL D'INFORMATIQUE

Le club MICROTEL LES LILAS et le Centre d'Echanges Internationaux organisent pour la troisième année 4 stages d'informatique, pour les 15-20 ans pendant les mois de juillet et août. Ces stages se déroulent dans un magnifique château en Touraine, en présence de groupes étrangers (américains, allemands, turcs...).

On aura la possibilité de s'initier également au tir à l'arc, au théâtre, et à l'équitation... Les stages sont réalisés sur des micro-ordinateurs compatibles IBM PC et durent 15 jours.

Du 5 juillet au 19 juillet, stage pascal ou assembleur

Du 19 juillet au 2 août, stage basic

Du 2 août au 16 août, stage basic

Du 16 août au 30 août, stage pascal ou assembleur.

Stage basic : 2740 F

(pension comprise).

Stage pascal ou assembleur : 2990

(pension comprise).

### RENSEIGNEMENTS :

CEI — 104, rue Vaugirard

75006 PARIS

Tél. (1) 45.49.26.25.

## CRÉATION DE CARACTÈRES

Envoyez vos programmes à Tremplin Micro.

**Qui veut se dévouer ? nous avons besoin de deux ou trois petits programmes.**

Plusieurs logiciels créent des fichiers de caractères destinés à être implantés sur l'imprimante Image Writer ou sur une des imprimantes Epson ayant l'option "téléchargement de caractères".

Comme de bien entendu, chaque auteur a travaillé dans son coin (à commencer par moi) et chaque fichier a une structure propre, qui l'empêche de passer sur les autres logiciels.

1. Vous trouverez ici la composition des fichiers créés par *Gribouille* pour *Image Writer*. Ce sont des fichiers binaires de 957 caractères, composés de 3 éléments :

A. Les 4 octets qui annoncent la création de caractères sur 8 points de large. Ces 4 octets sont Esc - Esc I (valeur en décimal : 27 45 27 73, en hexadécimal : \$1B \$2D \$1B \$49).

B. Une suite de 950 octets, correspondant à 95 caractères. Chacun est décrit sur 10 octets, qui sont :

— 1<sup>er</sup> octet, le code ASCII du caractère. Nous trouvons, de 10 en 10 octets, les valeurs 32 = espace (\$20 en hexadécimal) à 126 = tréma (\$7F en hexadécimal).

— 2<sup>e</sup> octet, la lettre H qui symbolise la largeur de 8 points. On trouvera un H majuscule (72 ou \$48)

si le caractère doit être imprimé en haut (cas le plus fréquent) et un h minuscule (104 ou \$68) s'il doit être en bas (exemple le g ou le q minuscules).

— Les 8 octets suivants ont des valeurs qui déterminent le dessin de chacune des colonnes de 8 points dont le caractère se compose. Zéro donne une colonne à blanc. \$AA donne une colonne en pointillé (10101010 en binaire : chaque 1 provoque l'impression d'un point).

C. Les 3 derniers octets sont :

4, qui annonce à l'imprimante la fin des caractères personnalisés.

Esc' (Escape apostrophe = 27 39 ou \$1B \$27) qui rend actifs les caractères personnalisés.

2. A vous de jouer : si vous avez des fichiers de caractères destinés à l'*Image Writer*, mais provenant de logiciels différents, écrivez (en Basic AppleSoft ou en Assembleur) les routines permettant de passer d'un fichier à l'autre. Il suffira peut-être d'ajouter ou de supprimer quelques octets en début ou en fin de fichier. Le problème vous paraîtra sûrement très simple, mais si vous savez le résoudre avec élégance, en offrant aux utilisateurs éventuels une bonne "interface", vous aurez fait œuvre utile.

Madeleine HODÉ, auteur de *Gribouille*.



# Quelques adresses intéressantes des APPLE IIe, IIc et GS

**\$FC9E (64670) CLEOLZ** Efface une ligne à partir de la position contenue dans Y. Y=0 correspond à HTAB 1.

```
10 VTAB1: HTAB1: PRINT "CECI EST SEULEMENT UN EXEMPLE";
20 CALL 768
```

La ligne sera effacée à partir du "T" de "EST"

```
300 : A0 07 LDY 07
302 : 4C 9E FC JMP FC9E
```

**\$FC36 (64566) CLRTOP** Au contraire de \$FC32 CLRSCR (64562) qui efface la totalité de l'écran GR, CLRTOP n'en efface que 40 lignes, si bien que la fenêtre texte reste intacte.  
CALL 64566 ou CALL - 970

**\$FD8B (64907) CROUT1** Efface la fin de la ligne et génère un RETURN. Ainsi, dans l'exemple ci-après, le curseur sera au début de la troisième ligne :

```
10 VTAB1: HTAB1: PRINT "CECI EST UN EXEMPLE";
20 CALL 64907: REM ou call - 629
```

**\$FC24 (64548) VTABZ** Déplace le curseur à une position qui doit être contenue dans l'accumulateur. Exemple :

```
10 HOME: FOR I = 10 TO 1 STEP - 1: VTAB I: PRINT I: NEXT
20 I=9: GOSUB 30: I=3: GOSUB 30: I=0: GOSUB 30: END
30 POKE 769,I: CALL 768: HTAB 4: PRINT"RECU": RETURN
```

```
300 : A9 00 LDA 00 Alimenté par le POKE 769,I du Basic
302 : 4C 24 FC JMP FC24 Place le curseur au début de la ligne A.
```

**\$FB5B (64347) TABV** Place la valeur de l'Accumulateur dans CV (\$25) et appelle \$FC22 VTAB qui déplace le curseur en CV.

```
300 : A9 05 LDA 05 Correspond à VTAB 6
302 : 4C 5B FB JMP FB5B Effectue le travail (et CV contient 5).
```

La ligne ci-après, associée aux deux instructions précédentes, provoquera l'affichage de "ESSAI" au début des lignes 6 et 20 :

```
10 HOME: VTAB20: PRINT "ESSAI": CALL 768: PRINT "ESSAI"
```

**\$FBF0 (64496) STORADV** Pour afficher les icônes, c'est parfait, mais en colonnes impaires (80 colonnes), et normal en 40 colonnes.

Attention ! il faut d'abord activer la carte 80 colonnes. Pour obtenir les icônes en 40 colonnes, taper ESC 4 PRINT CHR\$(17).

```
300 : A2 40 LDX 40 Pomme ouverte.
302 : 8A TXA X passe dans A
303 : 20 F0 FB JSR FBF0 Storadv (X et A sont détruits).
306 : E8 INX
307 : E0 60 CPX 60 Les icônes vont jusqu'à 65F.
309 : 90 F7 BCC 0302
30B : 60
```

Il suffirait de commencer la routine par 2C 55 C0 et de la terminer par 2C 54 C0 pour afficher dans les colonnes paires. ■

# UN BUG CORRIGÉ

Bonne nouvelle, les auteurs du *IIGS* ont corrigé un bug de l'APPLESOFT.

Un examen détaillé de la ROM du nouvel APPLE montre que l'on a repris la ROM ASOFT du *IIC* en y corrigeant un bug en GR. Dans le *IIC*, PLOT, HLIN et VLIN utilisaient la routine PLOTFNS en \$F1EC pour récupérer D1, D2 deux valeurs déterminant les paramètres nécessaires à ces routines.

Pour PLOT D1 = abscisse, D2 = ordonnée.

Pour HLIN D1 = abscisse1, D2 = abscisse2.

Pour VLIN D1 = ordonnée1, D2 = ordonnée2.

Dans le *IIE* PLOTFNS refusait  $D1 > 48$  et  $D2 > 48$ . C'était acceptable puisque GR n'avait que 40 colonnes. Dans le *IIC* on refusait  $D1 > 80$  et  $D2 > 48$  ce qui permettait un bon fonctionnement de PLOT et VLIN (avec un contrôle "lâche") mais pour HLIN ça ne collait pas (D2 limité à 48).

Dans le *IIGS* on a gardé PLOTFNS version *IIE*, et on l'utilise pour PLOT et VLIN qui DONC refusent  $D1 > 48$  ou  $D2 > 48$ . Pour HLIN on a écrit une nouvelle version de HLOTFNS implantée en \$F8D9 qui elle refuse  $D1 > 80$  ou  $D2 > 80$ .

\$F1EC <i>IIE</i> et <i>IIGS</i>		\$F1EC <i>IIC</i>	\$F8D9 <i>IIGS</i>		
PLOTFNS	JSR GETBYT	JSR GETBYT	GLOTFNS	JSR GETBYT	
	CPX £48	CPX £80		CPX £80	
	BCS GOERR	BCS GOERR		BCS GGOERR	
	STX FIRST	STX FIRST		STX FIRST	
	LDA £','	LDA £','			
	JSR SYNCHR	JSR SYNCHR		JSR CHKCOM	
	JSR GETBYT	JSR GETBYT		JSR GETBYT	
	CPX £48	CPX £48		CPX £80	
	BCS GOERR	BCS GOERR		BCS GGOERR	
	STX H2	STX H2		STX H2	
	STX V2	STX V2		STX V2	
	RTS	RTS		JMP \$F20C	...LINCOOR
GOERR	JMP IQERR	JMP IQERR	GGOERR	JMP GOERR	???????

Pourquoi tout cela me direz-vous ? Tout simplement parce que l'on peut dessiner en GR 80 colonnes ce qui permet par exemple de réaliser très facilement des présentations du genre histogramme.

Pour ce faire, il faut activer normalement le mode 80 colonnes, activer le mode DGR comme le mode DHGR par POKE 49278,0 (pour *IIC*) puis POKE 49246,0. Pour quitter faire POKE 49247,0, puis POKE 49279,0 et enfin TEXT.

Bien entendu le POKE 49234,0 classique vous donnerait un écran GR complet.

Je note au passage que les auteurs de cette dernière révision ont apparemment gaspillé 29 octets en ROM F8. Puisque les routines sont obligées de

faire leurs propres tests de validité sur D1 et D2 lorsqu'il s'agit de coordonnées horizontales, il me semble qu'il était plus logique de coder comme suit : (voir tableau de la page 42).

F220 est en fait la fin de l'actuelle routine LINCOOR. On y trouve : F220 CPX £48, BCS GOERR, RTS c'est tout ce qu'il nous fallait.

Bien évidemment le JSR GLOTFNS de la routine HLIN devrait être remplacé par un JSR PLOTFNS.

Si je ne m'abuse, tous les contrôles nécessaires sont faits et l'on conserve l'implantation et l'encombrement des routines originelles.

Pendant que j'y pense, et bien que cela n'ait rien à voir avec le GR, je vous signale... (suite page 42)

que contrairement au 6502, le 65816 considère, même en mode émulation le BRK comme une instruction sur 2 octets. Cela peut donner des résultats "bizarres" lors d'un désassemblage. Même "pro-

blème" avec le mini-assembleur qui exige désormais un opérande de un octet après le BRK. Si vous souhaitez mettre un unique 00 il faut utiliser :00 au lieu de BRK (c'est le mode garnissage de table).

PLOTFNS	JSR GETBYT	F209 LINCOOR	inchangée
	STX FIRST		
	JSR COMBYTE (\$E74C)	F225 PLOT	JSR NPLOT
	STX H2		suite inchangée
	STX V2	F241 VLIN	JSR LINCOOR
	RTS		TXA
NPLOT	JSR PLOTFNS		TAY
	JMP F220 (\$F220)		CPY £40
NVLIN	JSR F220		BCS GOERR
	TXA		LDX FIRST (changé)
	JMP VLIN		JMP NVLIN (changé)

Yvan KOENIG.

# TESTEZ GRGS !

```

100 TEXT :D$ = CHR$ (4): PRINT D$"PR£3": PRINT : GOSUB 175      627A
105 CALL 768: GOSUB 170                                          7EB3
110 POKE 49246,0                                                0C1E
115 POKE 49236,0: POKE 49234,0: POKE 49238,0: POKE 49232,0: GOSUB 170  9BA0
120 FOR I = 0 TO 47: COLOR= INT ( RND (1) * 15): HLIN 0,79 AT I: NEXT  BDDF
125 FOR J = 0 TO 79: COLOR= INT ( RND (1) * 15): VLIN 0,47 AT J: NEXT  74E2
130 FOR J = 79 TO 0 STEP - 1: COLOR= INT ( RND (1) * 15): VLIN 0,47 A
    T J: NEXT                                                    D3A3
135 FOR I = 47 TO 0 STEP - 1: COLOR= INT ( RND (1) * 15): HLIN 0,79 A
    T I: NEXT                                                    4DA0
140 IF PEEK (49249) < 128 THEN 120                                59AF
145 CALL 768                                                      8331
150 POKE 49247,0: POKE 49236,0: POKE 49233,0                    15CA
155 FOR I = 1 TO 3: VTAB 10 + I: HTAB 30: PRINT "                ";
    NEXT : VTAB 12: HTAB 31: PRINT "(M)ENU DISQUETTEctG ";: GET R$: PRI
    NT : CALL 768                                                D1EC
160 IF R$ = "M" OR R$ = "m" THEN PRINT D$"RUN MENU,D1"          753E
165 HOME : END                                                    8E51
170 CALL - 198: FOR I = 1 TO 100:X = PEEK (49200): NEXT : RETURN  1189
175 FOR I = 768 TO 794: READ R: POKE I,R: NEXT : RETURN          0918
180 DATA 162,23,138,32,71,248,160,39,169,78,145,38,44,85,192,145,38,44,
    84,192,136,16,243,202,16,232,96                               E898

```

# BON à SAVOIR

## TriDos

UTILITAIRE  
POUR APPLE IIc

Si vous possédez un Apple IIc (128 ou 384 Ko) et une imprimante ImageWriter, offrez-vous TriDos (version 2-3) en confiance. Cet utilitaire captera directement le catalogue de vos disquettes pour l'inclure dans sa propre base de données... quel que soit le système d'exploitation utilisé : Pascal, DOS 3.3 ou ProDOS.

**QUESTION :** Et si j'utilise un quatrième système ?

**RÉPONSE :** Vous vous contenterez alors de saisir manuellement la liste des fichiers de la disquette incriminée...

### Les points forts de TriDos :

- Simplicité de l'utilisation grâce à des menus parlants.
- Possibilité de supprimer des titres ou des disquettes, ou encore de modifier la liste des fichiers de la base de données ;
- Création facultative d'un index exploitable avec AppleWorks ;
- Fonction d'aide intégrée ;
- Mode d'emploi succincte, mais clair.

**SES LIMITES :** TriDos vous permettra d'enregistrer 200 disquettes et 500 titres différents. Cela paraît énorme, mais la limite est assez rapidement atteinte, ce qui est normal. Il est évidemment souhaitable de placer la base de données sur une disquette séparée.

**COMPATIBILITÉ :** Avec l'Apple IIe, mauvaise. Avec le GS, il existe un problème d'affichage (écran de présentation) facile à résoudre en modifiant une ligne du programme (non protégé) en Basic. Ensuite, il semble que les choses se passent normalement, mais Badaroux C.A. ne garantit pas cette compatibilité.

**CONCLUSION :** Quand on sait que TriDos n'est vendu que 180 F TTC on a bien envie de le commander les yeux fermés. On en sera pas déçu !

NESTOR

Badaroux C.A. 144, rue Legendre 75017 PARIS PARIS (vente directe, franco).



## TOUJOURS LE LANGAGE C

Franchement, si vous désirez, programmer en **C** — et je crois que vous y pensez sérieusement —, étudiez cette excellente **INTRODUCTION À LA PROGRAMMATION**.

Thomas Plum, président de la Société Plum Hall Inc. II, vous prouvera que le **C** peut devenir Votre langage préféré. Vous apprécierez la clarté des explications, mais aussi le nombre important des exemples.

C'est à mon avis l'une des meilleures introductions au **C**. Les conditions de portabilité y sont bien traitées, et pas seulement sur de gros systèmes. Un guide des instructions du langage et des bibliothèques offertes par UNIX est fourni en annexe. Ceci dit, il est évident que les utilisateurs d'Apple — et notamment du GS — n'y trouveront rien concernant Leur machine.

Par contre, après avoir potassé ce bouquin, ils seront fin prêts pour écrire des programmes géniaux !

"**C**" est à dire !

NESTOR

Inter Editions, 87, rue du Maine 75014 PARIS

(340 pages).

## MODIFIER.PRO

10 TEXT : HOME	1C5A
20 D\$ = CHR\$(4)	B3A4
30 PRINT D\$"BLOAD OBJ,A\$2A00"	4127
40 PRINT D\$"BRUN MODIFI"	DB75
50 PRINT D\$"BLOAD DESAS.PRO,A\$2E0D"	B2F3
60 PRINT D\$"BSAVE NEWOBJ,A\$2800,L4608"	2461
70 PRINT "Maintenant il faut changer l'adresse du fichier"	D677
80 PRINT "dans le directory principal avec votre DiskZAP"	5591
90 PRINT "L'adresse enregistrée \$2800 doit devenir \$9800"	03AB

## MODIFIER.DOS

10 TEXT : HOME	1C5A
20 D\$ = CHR\$(4)	B3A4
30 PRINT D\$"BLOAD SRCRR.OBJ,A\$4000"	72D2
40 PRINT D\$"BLOAD DESAS.DOS,A\$4DA1"	12E7
50 POKE 17993,133	3689
60 POKE 17994,151	048A
70 POKE 19526,164	ED87
80 POKE 19530,151	D37E
90 PRINT D\$"BSAVE SRCRR.OBJ.2GS,A\$4000,L\$10EC"	C060

## MODIFI

(BSAVE MODIFI,A\$300,L131)

0300: A0 00 84 3C A9 2A 85 3D A9 0C 85 3E A9 30 85 3F	860A
0310: 84 42 A9 28 85 43 20 2C FE A2 3F 8E 4A 28 A2 99	6CC5
0320: 8E 25 28 8E 2D 28 E8 8E 53 28 8E 41 31 8E 4C 31	85BA
0330: 8E 04 34 8E 94 34 E8 8E F0 33 E8 8E 66 30 8E EB	DA3A
0340: 30 8E 59 31 8E 73 31 8E B9 31 8E EB 36 8E F3 36	1A58
0350: E8 8E 69 28 8E 6C 28 8E FA 28 8E 4E 30 8E 80 30	4E23
0360: 8E B4 30 8E C8 30 8E FF 30 8E 6A 36 8E 95 37 8E	1BCB
0370: F7 38 A2 9F 8E DE 32 8E 03 33 A2 F1 8E DD 32 8E	1490
0380: 02 33 60	9695

## DESAS.PRO

(BSAVE DESAS.PRO,A\$2E0D,L512)

2E0D: A6 3A A4	6084
2E10: 3B 20 96 FD 20 48 F9 A1 3A A8 4A 90 05 6A B0 0C	12D7
2E20: 29 87 4A AA BD 4B 9F 20 79 F8 D0 04 A0 FC A9 00	A2F5
2E30: AA BD 8F 9F 85 2E 29 03 85 2F 20 DF 9F F0 18 29	29F7
2E40: 8F AA 98 A0 03 E0 8A F0 0B 4A 90 08 4A 4A 09 20	F278
2E50: 88 D0 FA C8 88 D0 F2 60 20 0D 9E 48 B1 3A 20 DA	74BC
2E60: FD A2 01 20 4A F9 C4 2F C8 90 F1 A2 03 C0 04 90	6C38
2E70: F2 68 A8 B9 CB 9E 85 2C B9 0B 9F 85 2D A9 00 A0	6E33
2E80: 05 06 2D 26 2C 2A 88 D0 F8 69 BF 20 ED FD CA D0	BF00
2E90: EC 20 48 F9 A4 2F A2 06 E0 03 F0 1C 06 2E 90 0E	F589
2EA0: BD A2 9F 20 ED FD BD 9C 9F F0 03 20 ED FD CA D0	BD97
2EB0: E7 60 88 30 E7 20 DA FD A5 2E C9 E8 B1 3A 90 F2	8BCE
2EC0: 20 56 F9 AA E8 D0 01 C8 4C 40 F9 1C 8A 1C 23 5D	9361
2ED0: 8B 1B A1 9D 8A 1D 23 9D 8B 1D A1 1C 29 19 AE 69	3409

# SOURCEROR

Par Yvan KOENIG

Je viens juste d'acheter un IIGS en remplacement de mon vieil APPLE II plus. Hélas ! les premières difficultés surgissent !

Un de mes outils favoris est l'assembleur MERLIN accompagné du désassembleur SOURCEROR. Pour MERLIN, pas de gros problème, si ce n'est que l'on doit passer par une carte interface imprimante pour se servir de la commande PUT. Sinon le port série semble utilisable.

Le vrai problème réside dans l'impossibilité d'utiliser SOURCEROR qui appelle une routine disparue LIST2. J'ai extrait d'un de mes anciens programmes le module DESAS légèrement remanié. Ce numéro de Tremplin étant quasiment complet, les fichiers sources figurent uniquement sur la disquette.

## UN REMÈDE POUR 2 VERSIONS

- Sous DOS 3.3 je traite le cas de BIG.MAC.C, tel qu'il m'a été vendu par CALL A.P.P.L.E.
- Sous ProDOS je traite le SOURCEROR/OBJ qui figure sur le disque portant la référence V2.52, achetée chez Roger Wagner.

## PRODOS

Placez sur une disquette le fichier SOURCEROR/OBJ, rebaptisé OBJ. Ajoutez MODIFIER.PRO, MODIFI et DESAS.PRO. Faites alors RUN MODIFIER.PRO. Le fichier OBJ sera chargé en \$2A00, coupé en deux en abaissant le début en \$2800.

31 octets seront ajustés. Le module DESAS.PRO sera placé dans le trou de 2 pages précédemment créé. Le tout sera sauvé sous le nom NEWOBJ. Tout cela se faisant en bas

# SUR IIGS

de la mémoire, il vous faudra corriger l'adresse du fichier en intervenant directement dans le directory à l'aide de votre programme de ZAP favori (en fait ce sera peut-être plutôt celui qui acceptera de travailler sur le IIGS car les ProZAP de NIBBLE et de CALL APPLE sont fâchés avec cette machine, tandis que celui de POM'S25 fonctionne). Quand tout cela sera réalisé, reportez NEWOBJ sous le nom SOURCEROR/OBJ sur une copie de MERLIN, soyez prudent.

## DOS 3.3 avec BIG.MAC.C

Sur une copie, renommez ASM.OBJ en ASM-Obj, ASM.OBJ//E en ASM.OBJ, SRCRR.OBJ en SRCRR-Obj, SRCRR.OBJ LC en SRCRR.OBJ puis, copiez SRCRR.OBJ sur un disque portant MODIFIER.DOS et DESAS.DOS.

RUN MODIFIER.DOS placera DESAS.DOS au début de la table des labels prédéfinis (qui seront donc perdus), corrigera 4 octets et sauvera le résultat sous le nom SRCRR.OBJ.2GS qu'il vous faudra transférer sur une COPIE de BIG.MAC.C sous le nom SRCRR.OBJ. Attention ! n'utilisez pas l'utilitaire LABELER pour modifier la table des labels avec ces patches en place.

A l'intention de ceux qui n'ont pas la disquette *Tremplin Micro*, je donne un tableau des 22 différences entre DESAS.PRO et DESAS.DOS.

DESAS	.PRO	.DOS	DESAS	.PRO	.DOS
2E25	4B	DF	2E7A	9F	96
2E26	9F	96	2EA1	A2	36
2E32	8F	23	2EA2	9F	97
2E33	9F	97	2EA7	9C	30
2E3B	DF	73	2EA8	9F	97
2E3C	9F	97	2FE3	A9	3D
2E59	0D	A1	2FE4	9F	97
2E5A	9E	95	2FEC	C8	5C
2E74	CB	5F	2FED	9F	97
2E75	9E	96	2FF2	58	EC
2E79	0B	9F	2FF3	9E	95

Bon courage et à bientôt. Y. KOENIG

2EE0:	A8 19 23 24 53 1B 23 24 53 19 A1 AD 1A 5B 5B A5	6FEC
2EF0:	69 24 24 AE AE AB AD 29 BA 7C 8B 15 9C 6D 9C A5	C67B
2F00:	69 29 53 84 13 34 11 A5 69 23 A0 D8 62 5A 48 26	9194
2F10:	62 94 88 54 44 C8 54 68 44 EB 94 C4 B4 08 84 74	77D2
2F20:	B4 28 6E 74 F4 CC 4A 72 F2 A4 8A 06 AA A2 A2 74	28C2
2F30:	74 74 72 44 68 B2 32 B2 72 22 72 1A 1A 26 26 72	0994
2F40:	72 88 C8 C4 CA 26 48 44 44 A2 C8 0F 22 FF 33 CB	E9DE
2F50:	62 FF 73 03 22 FF 33 CB 66 FF 77 0F 20 FF 33 CB	ADFE
2F60:	60 FF 70 0F 22 FF 39 CB 66 FF 7D 0B 22 FF 33 CB	BE0F
2F70:	A6 FF 73 11 22 FF 33 CB A6 FF 87 01 22 FF 33 CB	6A94
2F80:	60 FF 70 01 22 FF 33 CB 60 FF 70 24 31 65 78 00	51F0
2F90:	21 81 82 59 4D 91 92 86 4A 85 9D 49 5A D9 00 D8	9733
2FA0:	A4 A4 00 AC A9 AC A3 AB A4 12 14 1A 1C 32 34 3A	4C34
2FB0:	3C 52 5A 64 72 74 7A 7C 89 92 9C 9E B2 D2 F2 FC	5EEF
2FC0:	C3 AE C1 8A 8B A5 AC 00 38 FB 37 FB 39 21 36 21	FEAE
2FD0:	3A F8 FA 3B FA F9 22 21 3C FA FA 3D 3E 3F FC 98	D31B
2FE0:	A2 16 DD A9 9F F0 04 CA 10 F8 6D BD C8 9F A0 00	87C7
2FF0:	60 20 58 9E 20 53 F9 85 3A 84 3B A9 00 6D D9 F6	D738
3000:	E1 EE CB 74 74 76 C6 00 CF C5 CE C9 C7	2380

## DESAS.DOS

(BSAVE DESAS.DOS,A\$2E0D,L512)

2E0D:	A6 3A A4	6D84
2E10:	3B 20 96 FD 20 4B F9 A1 3A AB 4A 90 05 6A B0 0C	12D7
2E20:	29 87 4A AA BD DF 96 20 79 F8 D0 04 A0 FC A9 00	8A80
2E30:	AA BD 23 97 85 2E 29 03 85 2F 20 73 97 F0 18 29	900F
2E40:	8F AA 98 A0 03 E0 8A F0 0B 4A 90 08 4A 4A 09 20	F278
2E50:	88 D0 FA C8 88 D0 F2 60 20 A1 95 48 B1 3A 20 DA	8347
2E60:	FD A2 01 20 4A F9 C4 2F C8 90 F1 A2 03 C0 04 90	6C38
2E70:	F2 68 AB B9 5F 96 85 2C B9 9F 96 85 2D A9 00 A0	214A
2E80:	05 06 2D 26 2C 2A 88 D0 F8 69 BF 20 ED FD CA D0	BF00
2E90:	EC 20 48 F9 A4 2F A2 06 E0 03 F0 1C 06 2E 90 0E	F589
2EA0:	BD 36 97 20 ED FD BD 30 97 F0 03 20 ED FD CA D0	1AAF
2EB0:	E7 60 88 30 E7 20 DA FD A5 2E C9 E8 B1 3A 90 F2	8BCE
2EC0:	20 56 F9 AA E8 D0 01 C8 4C 40 F9 1C 8A 1C 23 5D	9361
2ED0:	8B 1B A1 9D 8A 1D 23 9D 8B 1D A1 1C 29 19 AE 69	3409
2EE0:	A8 19 23 24 53 1B 23 24 53 19 A1 AD 1A 5B 5B A5	6FEC
2EF0:	69 24 24 AE AE AB AD 29 BA 7C 8B 15 9C 6D 9C A5	C67B
2F00:	69 29 53 84 13 34 11 A5 69 23 A0 D8 62 5A 48 26	9194
2F10:	62 94 88 54 44 C8 54 68 44 EB 94 C4 B4 08 84 74	77D2
2F20:	B4 28 6E 74 F4 CC 4A 72 F2 A4 8A 06 AA A2 A2 74	28C2
2F30:	74 74 72 44 68 B2 32 B2 72 22 72 1A 1A 26 26 72	0994
2F40:	72 88 C8 C4 CA 26 48 44 44 A2 C8 0F 22 FF 33 CB	E9DE
2F50:	62 FF 73 03 22 FF 33 CB 66 FF 77 0F 20 FF 33 CB	ADFE
2F60:	60 FF 70 0F 22 FF 39 CB 66 FF 7D 0B 22 FF 33 CB	BE0F
2F70:	A6 FF 73 11 22 FF 33 CB A6 FF 87 01 22 FF 33 CB	6A94
2F80:	60 FF 70 01 22 FF 33 CB 60 FF 70 24 31 65 78 00	51F0
2F90:	21 81 82 59 4D 91 92 86 4A 85 9D 49 5A D9 00 D8	9733
2FA0:	A4 A4 00 AC A9 AC A3 AB A4 12 14 1A 1C 32 34 3A	4C34
2FB0:	3C 52 5A 64 72 74 7A 7C 89 92 9C 9E B2 D2 F2 FC	5EEF
2FC0:	C3 AE C1 8A 8B A5 AC 00 38 FB 37 FB 39 21 36 21	FEAE
2FD0:	3A F8 FA 3B FA F9 22 21 3C FA FA 3D 3E 3F FC 98	D31B
2FE0:	A2 16 DD 3D 97 F0 04 CA 10 F8 6D BD C8 9F A0 00	48DF
2FF0:	60 20 EC 95 20 53 F9 85 3A 84 3B A9 00 6D D9 F6	43C3
3000:	E1 EE CB 74 74 76 C6 00 CF C5 CE C9 C7	2380



## L'AUTEUR

Né le 29 novembre 1950 à Cleveland dans l'Ohio, Kevin O'DONNELL est l'aîné de 7 frères et sœurs. En 1966, toute la famille O'DONNELL part s'installer à Séoul en Corée du Sud. Kevin, diplômé de l'école américaine de Séoul en 1968, regagne les Etats-Unis et intègre l'Université de Yale dans le Connecticut. C'est à cette époque que la lecture de romans de Science-Fiction lui donne envie d'écrire ses propres nouvelles. Une licence de chinois en poche, il part pour Hong Kong en 1972, puis à Taiwan, en tant que professeur d'anglais dans un collège. Il vend alors ses premières nouvelles à la revue "Analog Science-Fiction" avant de regagner le Connecticut. Pendant trois ans, il exerce successivement une douzaine de jobs à mi-temps afin de s'investir à fond dans l'écriture. Il se marie en 1974 avec Lillian Kia Chou Tchang, une jeune Chinoise rencontrée à Yale. Il continue de vendre des histoires courtes à des revues de Science-Fiction ou de Fantastique et se lance enfin dans l'écriture de son premier roman, roman qui sera publié en 1979 sous le titre "Bander Snatch" aux éditions Bantam Books. De 1979 à 1983, il est rédacteur, puis directeur du magazine "Empire for the Science-Fiction writer", magazine s'adressant aux auteurs de romans de Science-Fiction et d'ouvrages d'anticipation. Depuis, il a publié sept autres romans dont quatre forment les premiers tomes de la célèbre série "The journeys of Mc Gill Feighan". Plusieurs de ces ouvrages ont été nominés pour le prestigieux prix américain "Nebula Award". Passionné d'informatique, il consacre aujourd'hui une partie de son temps à la formation au sein des entreprises, formation au traitement de texte et à la gestion de bases de données. Il vit à Sunny Valley en Californie.

# KEVIN O'DONNELL a reçu le prix MANNESMANN TALLY 1987

pour son roman **ORA : CLE**

## LE LIVRE

Année 2188 : un monde automatisé, informatisé, branché sur tous les réseaux, où la plupart des gens travaillent à domicile face à des consoles d'ordinateurs, font leurs courses de chez eux par transmateur, prennent l'air sur leur terrasse et se risquent éventuellement à rendre parfois visite à leur voisin d'immeuble.

C'est qu'il ne fait pas bon sortir en 2188. Il est interdit de se promener dans la campagne et même dans les rues, afin de laisser repousser la végétation dans l'espoir qu'elle absorbera la pollution. Des extraterrestres, les Dacs, intrus du système solaire depuis quelques années, guettent le passant. Ils ont la fâcheuse manie de se livrer à la chasse à l'humain, et lorsqu'on transgresse leur curieux code de la chasse, ils peuvent désintégrer un immeuble ou une région entière.

Mais en dehors de ces petits inconvénients, Aël Elcatrevain vit heureux avec son épouse Emdée Aussincante, entre son clavier, son holophone et les bonsaïs de sa terrasse. Jusqu'au jour où il manque d'être tué par un Dac, parce que son guette-Dac est en panne. Simple accident ? pas sûr. Et lorsque son transmateur explose, est-ce une coïncidence ? Un singulier Wef, qui est un as de la programmation, va l'aider à y voir clair. Aël, sous l'apparence d'un paisible historien, est en fait un membre actif de l'ORA : CLE (Opinions, Recherches, Avis : Consultants par Liaison Electronique), et comme tel, relié à ce puissant réseau par un implant situé derrière son oreille gauche. L'identité des consultants par liaison électronique, les CLE (prononcer Seeley) est un secret farouchement gardé. Qui peut avoir percé le secret de l'identité d'Aël et qui peut bien avoir des raisons de tuer un historien pacifique ?

Il y a dans l'air comme une odeur de coup d'Etat et quand Aël comprendra l'enjeu de la bataille, il commencera à regretter d'avoir été si curieux, et peut-être d'avoir tué un Dac.

## LA FONDATION

LA FONDATION MANNESMANN TALLY est une association créée par la société MANNESMANN TALLY, premier constructeur européen d'imprimantes et notamment fournisseur de l'Education Nationale dans le cadre du plan "Informatique Pour Tous". Cette association a pour buts :

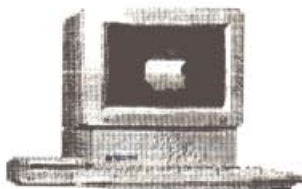
- la promotion et la diffusion des œuvres d'imagination mettant en scène l'informatique ;
- la création, la gestion et la mise en forme du règlement du PRIX MANNESMANN TALLY récompensant le meilleur ouvrage littéraire, récit ou roman, inspiré de l'informatique ;
- l'aide à la recherche scientifique vers les techniques dites de "l'Intelligence Artificielle".

Le bureau de la FONDATION est constitué de trois membres : Monsieur Dario ANGELINI (Directeur Général de MANNESMANN TALLY France en qualité de Président), Monsieur Jacques CHAMPAGNAC (Directeur Commercial en qualité de Trésorier), et Monsieur Thierry WELLHOFF (Directeur Général de l'agence WELLCOM en qualité de Secrétaire Général).

Les membres du jury du PRIX MANNESMANN TALLY sont membres d'honneur de droit de la FONDATION, le Président du jury est également Président d'honneur de la FONDATION. Pour la première année, le Bureau de la FONDATION a désigné Monsieur Bernard LENTERIC comme Président du jury. Le siège de la FONDATION est fixé 249 rue Saint-Jacques dans le 5ème, dans les bureaux de WELLCOM, agence conseil en communication de la société MANNESMANN TALLY.



APPLE II GS  
exclusivement



# TRANSBIN

Cette démonstration\* et la double routine LM qu'elle met en place nous montrent comment transférer un fichier binaire du banc 0 dans un autre banc (ici le 2), puis y lire les informations qu'il contient. Pour clarifier les procédures, nous utilisons au maximum le vieil Applesoft, ne serait-ce que pour démontrer qu'il peut encore s'accommoder (en association avec quelques octets de LM) d'une technologie aussi avancée que celle du GS.

\* Le fichier M6 est sensé contenir 3362 mots de 6 lettres, mais vous pouvez faire des essais avec une dizaine de mots (si le fichier des mots de 6 lettres vous intéresse, commandez tout simplement la disquette numéro 11 de *Tremplin Micro*).

## VARIABLES :

D : adresse à partir de laquelle le fichier est implanté (quel que soit ici le banc).

L : D + longueur du fichier - 6.

```

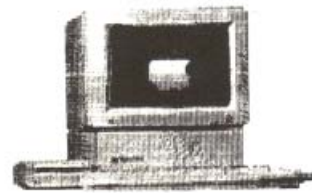
10 TEXT : NORMAL : D$ = CHR$ (4) : GOSUB 2
   00
15 PRINT D$"PR€3" : PRINT : HOME
20 INVERSE : PRINT " TRANSFERT D'UN FICHI
  ER BINAIRE DANS LA MEMOIRE HAUTE DE L'
  APPLE IIGS & LECTURE " : NORMAL
30 PRINT D$"BLOAD M6,A$2000"
40 CALL 768 : REM TRANSFERT EN 02/2000
50 D = 8192 : L = 28358 : POKE 8,2 : REM POUR
   02/D
60 POKE 6,D - INT (D / 256) * 256 : POKE
   7, INT (D / 256) : REM POUR../0000
70 CALL 793 : REM LECTURE ET AFFICHAGE
80 PRINT : GOSUB 190
90 IF PEEK (49152) = 27 THEN 110
100 D = D + 6 : IF D < L THEN 60
110 HOME : VTAB 22 : PRINT "(M)ENU DE DISQU
  ETTE €tG" : GET R$ : PRINT : IF R$ = "M
  " OR R$ = "m" THEN PRINT D$"RUN MENU"
190 CALL - 198 : POKE 49168,0 : WAIT 49152,
  128 : POKE 49168,0 : RETURN
200 FOR I = 768 TO 807 : READ R : POKE I,R :
  NEXT : RETURN
210 DATA 8,24,251,194,48,139,169,204,78,16
  2,0,32,160,0,32,84,2,0,171,226,48,56,2
  51,40,96,160,0,183,6,9,128,32,237,253,
  200,192,6,144,244,96

```

On dispose l'adresse du mot à lire dans les mémoires \$6-7-8 de la page zéro (ligne basic 60). L'indication du banc utilisé se trouve en \$8.



# Pour déplacer un bloc de données (en utilisant MVN), on doit avoir :



- Le nombre d'octets à transférer moins un (ici \$4ECB=20171) dans l'Accumulateur. MVN décrémentera l'accumulateur au fur et à mesure du transfert ; on aura A=\$FF à l'issue du transfert ;
- Les 16 bits de poids faibles de l'adresse-source (ici \$2000... du banc 01) dans le registre X ;
- Les 16 bits de poids faibles de l'adresse destinataire (c'est encore \$2000, mais du banc 02, dans notre exemple) dans le registre Y ;
- Le numéro du banc de destination dans le second octet de l'instruction MVN ;
- Le numéro du banc-source dans le troisième octet de l'instruction MVN.

Ainsi, **54 02 00** fait le transfert de **0** vers **2** et **54 0 02** transfère le bloc de **2** vers **0**.

## TRANSFERT (déplacement de bloc)

0=m    0=x    1=LCbank (0/1) A partir du Moniteur, tapez 0=m : 0=x, pour passer en mode natif et obtenir ce listage.

00/0300 :	08	PHP	Contenu du registre d'état empilé.
00/0301 :	18	CLC	Retenue à 0 + XCE = mode natif.
00/0302 :	FB	XCE	
00/0303 :	C2 30	REP £30	Commutation en mode natif pur (16 bits).
00/0305 :	8B	PHB	Registre DB (banc de données) empilé.
00/0306 :	A9 CB 4E	LDA £4ECC	
00/0309 :	A2 00 20	LDX £2000	Voir explication en bas de page.
00/030C :	A0 00 20	LDY £2000	
00/030F :	54 02 00	MVN 0002	Déplacement du bloc 0 en 02.
00/0312 :	AB	PLB	Récupération de DB (banc de données).
00/0313 :	E2 30	SEP £30	Commutation en mode natif mixte (8 bits).
00/0315 :	38	SEC	Retenue à 1 + XCE = mode émulation.
00/0316 :	FB	XCE	
00/0317 :	28	PLP	Récupération du registre d'état.
00/0318 :	60	RTS	Retour.

## LECTURE DANS UN AUTRE BANC

1=m    1=x    1=LCbank (0/1)

00/0319 :	A0 00	LDY £00	Register Y=0 (compteur).
00/031B :	B7 06	LDA °06\$.Y	Lecture de l'octet à l'adresse \$6-7-8 + Y.
00/031D :	09 80	ORA £80	Pour affichage normal.
00/031F :	20 ED FD	JSR FDED	COUT affiche le caractère.
00/0322 :	C8	INY	
00/0323 :	C0 06	CPY £06	On continue jusqu'à Y=6 (0 à 5).
00/0325 :	90 F4	BCC 031B é-0Cè	
00/0327 :	60	RTS	Retour.

# ALÉALIGNES

Voici un programme en assembleur qui permet de tracer des lignes joignant des points disposés aléatoirement sur un écran HGR. Il correspond à la demande d'un lecteur de *TREMLIN MICRO* (n°10, page 58). Roland JOST avoue n'en avoir pas bien saisi l'intérêt pratique, mais comme exercice de programmation, il en vaut bien un autre, pas vrai ?

ALÉALIGNES a le même effet que le programme BASIC suivant :

```
10 HGR2 : HCOLOR=3
20 X=INT(279 * RND(1)): Y=INT(191 * RND(1)):
  Hplot X,Y
30 X=INT(279 * RND(1)): Y=INT(191 * RND(1)):
  Hplot TO X,Y
40 IF PEEK(-16384) > 127 THEN END
50 GOTO 30
```

**Pour tester ALÉALIGNES :** HGR2 : CALL 768

Un appui sur une touche interrompt le tracé. On pourra le relancer par un nouveau CALL 768.

## QUELQUES COMMENTAIRES :

Tout d'abord une remarque importante : il est bien connu que la fonction RND de l'Applesoft ne fournit pas de vraies séries aléatoires. En effet, le plus souvent, après un certain temps la fonction tourne en rond. On peut s'en rendre compte par le fait que même après une durée assez longue, ALÉALIGNES ne remplit pas entièrement l'écran. On peut trouver des générateurs de nombres aléatoires plus performants dans les revues consacrées à l'Apple (*CALL APPLE, POM'S*, etc.), mais pour le problème présent, la fonction RND de l'Applesoft pourra toutefois convenir.

En APPLESOFT, l'appel d'un nombre aléatoire se fait par l'instruction  $Y=RND(X)$ , où l'argument X peut prendre les valeurs -1, 0 ou 1. Y sera un nombre réel compris entre 0 et 1. Donc pour obtenir un entier compris entre 0 et N, il faudra utiliser la fonction  $Y=INT(N * RND(X))$ . Si X est positif, un nombre différent est généré à chaque appel, si  $X=0$  on retrouve le dernier nombre généré, si X est négatif, le nombre

correspondant à X (stocké dans une table en mémoire) est fourni.

Le sous-programme RND réside en ROM à l'adresse \$EFAE. L'argument X doit se trouver dans le FAC (accumulateur flottant) avant l'appel de RND. En \$EFAE se trouve un appel à \$EFB2 qui teste le signe de l'argument X. Nous simplifierons le processus en chargeant 1 dans A avant de sauter en \$EFB1, court-circuitant ainsi le test du signe. Après exécution de RND, un nombre aléatoire compris entre 0 et 1 se trouve dans le FAC. Pour multiplier ce nombre par 279 (pour les abscisses) ou 191 (pour les ordonnées), on range d'abord FAC dans le deuxième registre flottant ARG par un appel à MOVAF (\$EB63). Puis la valeur 279 (ou 191) est mise dans le FAC : octet haut dans A, octet bas dans Y et appel de GIVAYF (\$E2F2) qui transforme un entier dans (Y,A) en flottant dans FAC.

FMULTT (\$E982) effectue la multiplication de FAC par ARG. FAC est enfin converti en entier sur 2 octets (LINUX et LINUX+1) par GETADR (\$E752). Enfin on transfère le contenu de LINUX dans XCOORH (\$06) et XCOORL (\$07).

Le tracé se fera simplement par les deux routines suivantes :

- **Hplot** : pour tracer un point aux coordonnées x,y il suffit de mettre l'octet bas de x dans X, l'octet haut dans Y et l'ordonnée y dans A, puis appeler \$F457.
- **Hplot TO x,y** : charger l'octet bas de x dans A, l'octet haut de x dans X et l'ordonnée y dans Y et enfin appeler \$F53A.

Pour faire bonne mesure, nous avons ajouté un effet sonore. Celui-ci peut être supprimé par 3 POKES : POKE 788,234: POKE 789,234: POKE 790,234. (suite page 50)

# ASSEMBLEUR EDASM PRODOS

Si vous ne possédez pas d'assembleur, contentez-vous de taper les codes inscrits en rouge, comme indiqué page 59.

## SAUVEGARDE :

BSAVE  
ALEALIGNES.0,  
A\$300,L\$73

```

1  * Tracé aléatoire de lignes en mode HGR
2  * R.Jost - 1986
3  *
4  *
5  * adresses en page zéro
6  *
7  XCOORDL EQU $06 ; octet bas de l'abscisse
8  XCOORDH EQU $07 ; octet haut de l'abscisse
9  YCOORD EQU $08 ; ordonnée
10 LINNUM EQU $50 ; registre 16 bits
11 *
12 * sous-routines Applesoft et Moniteur
13 *
14 STXTPT EQU $D697 ; réinitialise TXTPTR
15 GIVAYF EQU $E2F2 ; A,Y -> flottant dans FAC
16 SGNFLT EQU $E301 ; rend flottant l'entier dans Y
17 CONINT EQU $E6FB ; FAC -> X
18 FMULTT EQU $E982 ; multiplie ARG par FAC
19 GETADR EQU $E752 ; FAC -> entier dans LINNUM
20 MOVAF EQU $EB63 ; FAC -> ARG
21 ENRND EQU $EFB1 ; entrée non classique dans RND
22 HPLOT EQU $F457 ; trace un point HGR
23 HPLOTTO EQU $F53A ; trace une ligne HGR
24 BELL EQU $FBE4 ; émission d'un 'bip'
25 KBD EQU $C000 ; saisie au clavier
26 KBDSTRB EQU $C010 ; strobe
27 *
28 ORG $300
29 *
30 * initialisation
31 *
0300: A9 40 32 LDA $40
0302: 85 E6 33 STA $E6 ; page graphique 2
0304: A9 FF 34 LDA $FF
0306: 85 E4 35 STA $E4 ; couleur = blanc
36 *
37 * tracé du premier point
38 *
0308: 20 2B 03 39 JSR MUL280 ; X=INT(279*RND(1))
030B: 20 49 03 40 JSR MUL192 ; Y=INT(191*RND(1))
030E: 20 5F 03 41 JSR TRACE ; HPLOT X,Y
42 *
43 * boucle principale
44 *
0311: 20 2B 03 45 BOUCLE JSR MUL280 ; X=INT(279*RND(1))
0314: 20 E4 FB 46 JSR BELL ; un peu de bruit ..
0317: 20 97 D6 47 JSR STXTPT ; absolument nécessaire:
48 ; essayez donc de l'enlever!
031A: 20 49 03 49 JSR MUL192 ; X=INT(191*RND(1))
031D: 20 69 03 50 JSR TRACETO ; HPLOT TO X,Y
0320: AD 00 C0 51 LDA KBD ; teste le clavier
0323: 2C 10 C0 52 BIT KBDSTRB ; reprise lecture clavier
0326: C9 80 53 CMP $80 ; si pas de touche pressée
0328: 30 E7 54 BMI BOUCLE ; on recommence ...
032A: 60 55 RTS ; sinon retour au BASIC.

```

```

56 *
57 * génère un nombre aléatoire entre 0 et 1
58 * le multiplie par 279
59 * et sauve FAC dans XCOORL et XCOORH
60 *
032B: A9 01      61 MUL280  LDA  £01          ; argument de RND = 1
032D: 20 B1 EF  62          JSR  ENRND       ; entrée non classique ds RND
0330: 20 63 EB  63          JSR  MOVAF        ; FAC dans ARG
0333: A0 17      64          LDY  £$17         ; 279 dans Y
0335: A9 01      65          LDA  £$01         ; et A.
0337: 20 F2 E2  66          JSR  GIVAYF        ; transfère dans FAC
033A: 20 82 E9  67          JSR  FMULTT       ; multiplie ARG et FAC
033D: 20 52 E7  68          JSR  GETADR        ; FAC -> entier dans LINNUM
0340: A5 50      69          LDA  LINNUM        ; stocke
0342: 85 06      70          STA  XCOORL       ; dans XCOORL
0344: A5 51      71          LDA  LINNUM+1     ; et
0346: 85 07      72          STA  XCOORH       ; dans XCOORH
0348: 60        73          RTS
74 *
75 * génère un nombre aléatoire entre 0 et 1
76 * le multiplie par 191
77 * et stocke le résultat dans YCOOR
78 *
0349: A0 01      79 MUL192  LDY  £01          ; argument de RND = 1
034B: 20 B1 EF  80          JSR  ENRND       ; appel RND
034E: 20 63 EB  81          JSR  MOVAF        ; FAC dans ARG
0351: A0 BF      82          LDY  £$BF         ; 191
0353: 20 01 E3  83          JSR  SGNFLT       ; dans FAC
0356: 20 82 E9  84          JSR  FMULTT       ; multiplie FAC par ARG
0359: 20 FB E6  85          JSR  CONINT       ; FAC -> entier dans X
035C: 86 08      86          STX  YCOOR        ; sauve dans YCOOR.
035E: 60        87          RTS
88 *
89 * trace un point en haute résolution
90 *
035F: A5 08      91 TRACE   LDA  YCOOR        ; on charge les
0361: A6 06      92          LDX  XCOORL       ; registres
0363: A4 07      93          LDY  XCOORH       ; avant le tracé
0365: 20 57 F4  94          JSR  HPLOTT       ; du point.
0368: 60        95          RTS
96 *
97 * trace une ligne en haute résolution
98 *
0369: A5 06      99 TRACETO LDA  XCOORL       ; on charge les
036B: A6 07     100          LDX  XCOORH       ; registres
036D: A4 08     101          LDY  YCOOR        ; avant le tracé
036F: 20 3A F5  102          JSR  HPLOTTO      ; de la droite.
0372: 60     103          RTS

```

```

-BELL.....$FBE4      -BOUCLE....$0311      -CONINT....$E6FB      -ENRND.....$EFB1
-FMULTT....$E982     -GETADR....$E752     -GIVAYF....$E2F2     -HPLOTT....$F457
-HPLOTT0...$F53A     -KBD.....$C000      -KBDSTRB...$C010     -LINNUM....$0050
-MOVAF.....$EB63     -MUL192....$0349     -MUL280....$032B     -SGNFLT....$E301
-STXTPT....$D697     -TRACE....$035F     -TRACETO...$0369     -XCOORH....$0007
-XCOORL....$0006     -YCOOR....$0008

```

# AMORTISSEMENT

Comment calculer un amortissement linéaire ou dégressif qui respecte la réglementation fiscale ? Tout simplement en utilisant le programme en Basic de notre ami Robert CAZENAVE.

A vous de jouer... avec le clavier d'abord, puis avec les chiffres !

```

100 REM AMORTISSEMENT VERSION 1/86
105 : 003A
110 TEXT : HOME : CLEAR : PRINT CHR$(4);"PR&3": PRINT C  A3F7
    HR$(17); 003A
115 :
120 REM Changement du curseur
125 POKE 2043,223 DD45
130 : 003A
135 REM Arrondi avec 2 chiffres après la virgule
140 : 003A
145 DEF FN AR(X) = INT (100 * X + .5) / 100 D4E4
150 : 003A
155 DIM AN(17),UR(17) BD5B
160 PRINT TAB(10);"CALCUL DES AMORTISSEMENTS" BB91
165 PRINT CHR$(15); CHR$(27) 2C34
170 FOR K = 1 TO 10: PRINT "SSSS";: NEXT F36A
175 PRINT CHR$(14); CHR$(24) 1A30
180 VTAB 10: PRINT "1- AMORTISSEMENT LINEAIRE" D12D
185 VTAB 12: PRINT "2- AMORTISSEMENT DEGRESSIF" DB83
190 VTAB 14: PRINT "3- MENU" A814
195 VTAB 20: PRINT "VOTRE CHOIX:";: GET R$ F84A
200 IF R$ < > "1" AND R$ < > "2" AND R$ < > "3" THEN P 1098
    RINT CHR$(7);: HTAB 1: GOTO 195
205 IF R$ = "2" THEN AN$ = "AMORTISSEMENT DEGRESSIF": GOSU  BD9D
    B 235: GOTO 315
210 IF R$ = "1" THEN AN$ = "AMORTISSEMENT LINEAIRE": GOSUB  F54C
    235: GOTO 435
215 PRINT CHR$(4);"RUN STARTUP" B4ED
220 : 003A
225 REM Saisie des éléments de calcul
230 : 003A
235 HOME : INVERSE : PRINT AN$: NORMAL : VTAB 4: INPUT "BA  C677
    SE AMORTISSABLE:";B

```

240	VTAB 6: INPUT "DUREE (MAXIMUM 15 ANS):";DU: IF DU > 15	8ECD
	THEN HTAB 1: PRINT CHR\$(7);: GOTO 240	E373
245	TX = FN AR(100 / DU) / 100: REM TAUX LINEAIRE	8DC6
250	IF R\$ = "1" THEN 280	0E02
255	VTAB 8: INPUT "COEFFICIENT:";CO	D7C1
260	TD = (TX * CO)	6A03
265	VTAB 10	7E30
270	PRINT "TAUX DE L'AMORTISSEMENT DEGRESSIF:";TD	A2CC
275	IF R\$ = "2" THEN 285	
280	VTAB 12: INPUT "JOUR D'ACQUISITION:";JO: IF JO = 31 TH	7B6D
	EN JO = 30	
285	VTAB 14: INPUT "MOIS D'ACQUISITION (1 à 12) :";PR: IF	A859
	PR > 12 THEN PRINT CHR\$(7);: HTAB 1: GOTO 285	D778
290	VTAB 16: INPUT "ANNEE 19";A	E6E7
295	IF R\$ = "2" THEN PR = 13 - PR: GOTO 310	9442
300	NJ = (12 - PR) * 30 + 31 - JO	003A
305	:	63B1
310	RETURN	F6AD
315	D = DU	13FB
320	DU = DU + 1	03D2
325	FOR K = 1 TO D	315B
330	IF K > 1 THEN 375	003A
335	:	
340	REM Première annuité	003A
345	:	8516
350	AN(1) = FN AR((B * TD) * PR / 12)	F23B
355	VR(1) = FN AR(B - AN(1)): GOTO 415	003A
360	:	
365	REM Autres annuités	003A
370	:	
375	IF VR(K - 1) * TD > VR(K - 1) / (DU - K) THEN AN(K) =	7407
	VR(K - 1) * TD: GOTO 385	F602
380	AN(K) = VR(K - 1) / (DU - K)	
385	AN(K) = FN AR(AN(K)):VR(K) = FN AR(VR(K - 1) - AN(K)	21F2
	)	003A
390	:	003A
395	:	
400	REM Dernière valeur résiduelle	003A
405	:	D331
410	IF K = 10 THEN VR(K) = 0	
415	REM	
420	T = T + AN(K): NEXT	1E27
425	GOSUB 560	E74B
430	:	003A
435	REM Amortissement linéaire	
440	:	
445	NORMAL	003A
450	REM 1ERE ANNUITE	3B9D
455	IF NJ < > 360 THEN DU = DU +	
460	D = DU	7F3D
465	FOR K = 1 TO DU	F6AD
470	IF K > 1 THEN 520	AF27
475	:	6853
		003A

SI ON DEPASSE 1999. LE PROGRAMME  
SE PLANTE JOLIMENT. CORRIGER AINSI:

581	DA = 19:D2\$ = RIGHT\$(STR\$(A + K - 1),2): IF A + K - 1	1E27
	> 99 THEN DA = 20	E74B
585	PRINT " "DA;D2\$;: HTAB 15.-	003A
	LEN (STR\$(INT(AN(K)))):	
	PRINT AN(K);: HTAB 30 - LEN	003A
	(STR\$(INT(VR(K)))): PRINT	3B9D
	VR(K)	7F3D

(suite page 54)

## AMORTISSEMENT

(suite et fin)

```
480 : 003A
485 REM Première annuité
490 : 003A
495 AN(1) = FN AR((B * TX) * NJ / 360) 5556
500 VR(1) = FN AR(B - AN(1)): GOTO 535 D03E
505 : 003A
510 REM Autres annuités
515 : 003A
520 AN(K) = FN AR(B * TX):VR(K) = FN AR(VR(K - 1) - AN(K) 7D7F
    ))
525 REM Dernière annuité
527 : 003A
530 IF K = DU THEN AN(K) = FN AR(VR(K - 1)):VR(K) = 0 1D82
535 NEXT : GOSUB 560: END 98C1
540 : 003A
545 : 003A
550 REM Affichage
555 : 003A
560 HOME : INVERSE : PRINT AN#:; NORMAL : PRINT 1A1C
565 PRINT "BASE:";B;: HTAB 15: PRINT "T. LIN.:";TX;: HTAB
    28: PRINT "T. DEG: ";TD DB99
570 VTAB 4: HTAB 10: PRINT "ANNUITE";: HTAB 23: PRINT "VAL
    . RESID." E9D1
575 FOR K = 1 TO D 03D2
580 VTAB 5 + K A2EA
585 PRINT " 19";A + K - 1;: HTAB 15 - LEN ( STR# ( INT (A
    N(K))) ): PRINT AN(K);: HTAB 30 - LEN ( STR# ( INT (VR
    (K))) ): PRINT VR(K) 7962
590 NEXT 0582
595 POP 43A1
600 : 003A
605 REM Caractère souris
610 : 003A
615 PRINT CHR# (15); CHR# (27) 2C34
620 FOR K = 5 TO 20: VTAB K: HTAB 7: PRINT "Z";: HTAB 20:
    PRINT "Z";: HTAB 38: PRINT "Z": NEXT E277
625 PRINT CHR# (14); CHR# (24) 1A30
630 VTAB 22: PRINT "APPUYER SUR UNE TOUCHE";: GET R#: RUN 1FFD
```

## APPLE // PRODOS GUIDE DU PROGRAMMEUR

• par Marcel COTTINI

Une édition de *TREMLIN MICRO* indispensable à tous les programmeurs Apple-soft désirant tirer profit des nombreuses ressources d'un système qui a maintenant fait ses preuves : **ProDOS**. Prix TTC : 120 F

(Bulletin de commande page 75)



# LA SOURIS ET L'ASSEMBLEUR

## DEUXIÈME PARTIE : Initialisation

*Pour ceux d'entre vous que la première partie a intéressés (et qui ont bien révisé leur Assembleur), je n'ai qu'une chose à vous dire : "Tous à vos claviers !".*

### Voyons en premier lieu les étapes :

- Vérification de la présence de la carte Souris
- Mise en place du vecteur d'interruption
- Valeurs par défaut et synchronisation (INITMOUSE)
- Mise en service de la Souris et détermination du Mode (SETMOUSE).

Le programme se trouvant dans les pages suivantes est exécutable et se lance par CALL 24576 à partir du Basic ou \$6000G sous Moniteur, mais ne donnera rien de significatif à l'écran pour le moment, car il faudra attendre l'ajout des modules décrits dans les articles suivants. L'appel à la routine de traitement (INTERUP) a pourtant bien lieu 60 fois par seconde (vérifiez-le en faisant un CALL - 198 ou \$FF3AG et vous entendrez un bip particulièrement ralenti, preuve que le 6502 est occupé à autre chose).

Malgré la débauche de commentaires, un point particulier a pu vous paraître obscur. IL s'agit de cette histoire d'offset dans CALLCARD (offset ? Ké sek ça ?). Un offset peut se résumer à un écart ou une différence. En effet, il faut savoir que les microprogrammes Souris ne sont pas appelés directement par leur adresse absolue. On utilise pour ce faire une table, implantée à partir de \$Cn12 ("n" étant le numéro du connecteur où se trouve la carte Souris), dont chaque adresse contient un octet représentant l'octet de poids faible de l'adresse réelle, l'octet de

poids fort étant pour sa part implicite car étant toujours égal à \$Cn (\$C4 si slot n°4).

### Voici la table en question :

\$Cn12 :	Contient l'octet de poids faible de l'adresse de début de SETMOUSE
\$Cn13 :	SERVEMOUSE
\$Cn14 :	READMOUSE
\$Cn15 :	CLEARMOUSE
\$Cn16 :	POSMOUSE
\$Cn17 :	CLAMPMOUSE
\$Cn18 :	HOMEMOUSE
\$Cn19 :	INITMOUSE

### Que fait donc CALLCARD ?

Cette routine est chargée d'aller récupérer l'octet de poids faible de l'adresse réelle (cette opération s'effectue à la ligne 47), et après avoir vectorisé TOCARD (lignes 50 et 51), de passer à l'exécution de la routine concernée (ligne 53).

### Explication de la ligne 47 :

Le contenu de TMP n'est pas modifié dans le cours du programme et reste donc à \$00. On ajoute au contenu de TMP le registre Y contenant le code du microprogramme à exécuter, pour obtenir l'offset de position dans la table ci-dessus. L'accumulateur est alors chargé avec le contenu de l'adresse obtenue.

### Exemple avec INITMOUSE :

$$(TMP) + (Y) = \$00 + \$19 = \$19$$

La retenue éventuelle (si le résultat est supérieur à 255) est ajoutée au contenu de la mémoire suivante (adressage indexé indirect) :

(suite page 56)



$(TMP + 1) + Carry = \$Cn + 0 = \$Cn$  (dans le cas présent)  
 enfin :  $(\$Cn19) = \$1C$  (Vérifiez-le sous Moniteur)  
 d'où l'adresse réelle de début d'INITMOUSE :  $\$Cn1C$  ( $\$C41C$  si slot n°4).

Et voilà, c'est terminé pour aujourd'hui (pas trop mal au crâne ?). La prochaine fois nous verrons :

- Le traitement de l'interruption.
  - L'établissement des limites d'excursion de la Souris.
- Amusez-vous bien... François GALLET

**NOTA :** Si vous écrivez directement sous Moniteur, sauvez ce module par : **BSAVE SOURIS.C,A\$6000,L236**

```

0 *****
1 *
2 *   Programmation de la Souris : Deuxième partie *
3 *
4 *****
5 *
6 *
7 BELL      EQU   $FF3A      :Emet un bip
8 CH        EQU   $24        :Stockage du HTAB
9 CLS       EQU   $FC58      :Effacement de l'écran
10 CV       EQU   $25        :Stockage du VTAB
11 INIT     EQU   $19        :Initialise les valeurs par défaut de la souris
12 RDKEY    EQU   $FDOC      :Attente de la frappe d'une touche (GET)
13 SERVE    EQU   $13        :Déteçte si la Souris à causée l'interruption
14 SET      EQU   $12        :Initialise la souris avec le mode adéquat
15 STROUT   EQU   $DB3A      :Affiche une chaîne de caractères
16 TMP      EQU   $06        :Pointeur temporaire
17 VECTINT   EQU   $3FE      :Vecteur d'interruption
18 VTABZ    EQU   $FC24      :Effectue le VTAB
19 *
20          ORG   $6000
21 *
22 *   Début du programme
23 *   -----
24 *
25          SEI          :Interdit les interruptions
26          JSR   CHECK   ;Cherche la carte souris
27          LDA   £<INTERUP ;LB de l'adresse du programme d'interruption
28          LDX   £>INTERUP ;HB
29          STA   VECTINT  ;LB du vecteur interruption
30          STX   VECTINT+1 ;HB
31          LDY   £INIT    ;Offset de INITMOUSE
32          JSR   CALLCARD ;...
33          LDY   £SET     ;Offset de SETMOUSE et...
34          LDA   £$09     ;...mode de sélection des interruptions
35                   ;à chaque rafraichissement de l'écran
36          JSR   CALLCARD ;...
37 *
38          DS    24,234   ;Réserve d'octets (NOP) pour la suite
39 *
40          CLI          ;Rétablit les interruptions
41          RTS         ;Retour au basic
42 *
43 *   Sous programme : CALLCARD
44 *   -----

```

```

45 *
46 CALLCARD PHA           ;Sauve le contexte (valeur du Mode si SETMOUSE)
47     LDA (TMP),Y       ;Recherche de l'offset
48     LDX CN           ;Octet haut de l'adresse de la carte souris
49     LDY NO           ;...et n° du slot x 16
50     STA TOCARD+1     ;...vectorisé
51     STX TOCARD+2     ;pour saut indirect
52     PLA             ;Récupère le contexte (valeur du Mode si SETMOUSE)
53     JSR TOCARD       ;Saut indirect à la carte pour exécution...
54     RTS             ;et retour
55 *
56 *   Vectorisation
57 *   -----
58 *
59 CN     HEX 00         ;Stockage de l'octet de poids fort
60 NO     HEX 00         ;...et du n° de slot x 16
61 TOCARD JMP $0000     ;Opérande modifié par le programme
62 *
63 *   Sous programme : recherche du slot
64 *   -----
65 *
66 CHECK  LDX £$07       ;On teste 7 slots
67     LDA £$00         ;On commence la recherche
68     STA TMP         ;...aux adresses hautes
69     LDA £$C8         ;On commence par...
70     STA TMP+1       ;...
71 LOOP1  DEC TMP+1     ;le slot n°7
72     DEX             ;Compte les slots
73     BPL CONT        ;Si pas d'erreur on saute
74     LDA £<SOURIS    ;Sinon on affiche...
75     STA MESS+1     ;...le...
76     LDA £>SOURIS    ;...message...
77     STA MESS+3     ;d'erreur
78     BRA ERREUR     ;...
79 *
80 CONT   LDY £$0C       ;1er octet de signature
81     LDA (TMP),Y     ;...doit être là
82     CMP £$20       ;...et égal à $20
83     BNE LOOP1      ;Pas là , alors slot suivant
84     LDY £$FB       ;2ème octet de signature
85     LDA (TMP),Y     ;...en $CnFB
86     CMP £$D6       ;...égal à $D6 ?
87     BNE LOOP1      ;Non , alors slot suivant
88     LDA TMP+1     ;Oui , on sauve l'adresse
89     STA TOCARD+2   ;...pour plus tard (saut indirect)
90     STA CN         ;Utilisé par CALLCARD
91     ASL           ;On décale 4 fois...
92     ASL           ;...afin de faire passer...
93     ASL           ;...les 4 bits de poids faible à la place...
94     ASL           ;...des 4 bits de poids fort...
95     STA NO         ;pour obtenir le n° du slot x 16
96     ;ex : $C4 = 10100100 devient 01000000 = $40 ce
97     ;n'est autre que le slot n°4 multiplié par 16

```

(suite page 58)

```

98          ;(4 x 16 = 64 ou $40)
99          RTS          ;Retour d'appel
100 *
101 *
102 *   Affichage du message d'erreur
103 *   -----
104 *
105 ERREUR   PLA          ;Enlève l'adresse de retour de l'appel de CHECK
106         PLA          ;...
107 *
108         JSR   CLS      ;On efface l'écran
109         JSR   BELL     ;...et on émet un joli petit bip !
110         LDA   £$0C     ;Ligne du message d'erreur
111         STA   CV       ;...
112         JSR   VTABZ    ;Positionne le curseur à la ligne indiquée
113         LDA   £$14     ;HTAB
114         STA   CH       ;...
115 MESS     LDA   £$FF     ;Octet bas de l'adresse du message d'erreur
116         LDY   £$FF     ;_____ haut _____
117         JSR   STROUT   ; --> Affichage du message d'erreur
118         LDA   £$0C     ;Repositionne le
119         STA   CV       ;...VTAB
120         JSR   VTABZ    ;...
121         LDA   £$3A     ;...et le
122         STA   CH       ;HTAB
123 *
124         JSR   RDKEY    ;Attend un l'appui d'une touche
125 *
126         JSR   CLS      ;On efface l'écran...
127         CLI          ;...on rétablit les interruptions...
128         RTS          ;et on s'en retourne au Basic !
129 *
130 *
131 *   Message d'erreur
132 *   -----
133 *
134 SOURIS   ASC   "Pas de carte souris ---> Terminé !"
135         HEX   00
136 *
137 INTERUP  EQU   *          ;La prochaine fois , vous aurez la suite !
138         SEI          ;On interdit les interruptions nouvelles
139 *
140         PHA          ;Sauvegarde...
141         PHX          ;...du...
142         PHY          ;contexte
143 *
144         LDY   £SERVE   ;Offset de SERVEMOUSE
145         JSR   CALLCARD ;...
146         BCC   CLICK    ;Si Carry = 0 , la Souris est bien la cause de
147         ;l'interruption
148         JMP   RETINT   ;...sinon , retour
149 *
150 CLICK    NOP          ;Provisoirement !

```

```

151 *
152 RETINT PLY :Restauration...
153 PLX :...du...
154 PLA :contexte
155 *
156 CLI :Rétablit les interruptions...
157 RTI :et retour
158 *

```



Si vous n'êtes pas rompu(e) à la pratique de l'assembleur, voici les codes de la routine de François GALLET. Pour les taper, adoptez la marche à suivre habituelle : **CALL - 151**, suivi de **RETURN**, puis — à la suite de l'astérisque — chacune des lignes suivantes (ne pas oublier le **RETURN**, en fin de ligne). Attention ! ne tapez pas les valeurs hexa en couleur. Si vous disposez de notre disquette **SIGNATURE**, elles vous permettront de vérifier l'exactitude de la saisie :

**SAUVEGARDE : BSAVE SOURIS.II.C,A\$6000,L\$EC**

```

6000: 78 20 4D 60 A9 D8 A2 60 8D FE 03 8E FF 03 A0 19 619F
6010: 20 34 60 A0 12 A9 09 20 34 60 EA EA EA EA EA EA 0748
6020: EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA 83A0
6030: EA EA 58 60 48 B1 06 AE 48 60 AC 49 60 8D 4B 60 C96E
6040: 8E 4C 60 68 20 4A 60 60 00 00 4C 00 00 A2 07 A9 386A
6050: 00 85 06 A9 C8 85 07 C6 07 CA 10 0C A9 B5 8D 9C E5C2
6060: 60 A9 60 8D 9E 60 80 20 A0 0C B1 06 C9 20 D0 E7 6197
6070: A0 FB B1 06 C9 D6 D0 DF A5 07 8D 4C 60 8D 48 60 28BA
6080: 0A 0A 0A 0A 8D 49 60 60 68 68 20 58 FC 20 3A FF C95B
6090: A9 0C 85 25 20 24 FC A9 14 85 24 A9 FF A0 FF 20 046C
60A0: 3A DB A9 0C 85 25 20 24 FC A9 3A 85 24 20 0C FD BF69
60B0: 20 58 FC 58 60 D0 E1 F3 A0 E4 E5 A0 E3 E1 F2 F4 CF83
60C0: E5 A0 F3 EF F5 F2 E9 F3 A0 AD AD AD BE A0 D4 E5 1AE8
60D0: F2 ED E9 EE FB A0 A1 00 78 48 DA 5A A0 13 20 34 B8ED
60E0: 60 90 03 4C E7 60 EA 7A FA 68 58 40 EFE4

```

## INDISPENSABLES pour utiliser votre Apple :

- **CLINS D'OEIL AU 6502 DE L'APPLE** (avec disquette)
- **ROUTINES LM POUR 6502 ET 65C02** (avec disquette)
- **NOUVELLES ROUTINES POUR LE 65C02** (avec disquette)

Ces recueils regroupent des programmes en langage machine publiés dans *Tremplin Micro*, mais présentent aussi des routines originales ou remaniées, faciles à mettre en œuvre à partir du Basic, et permettant d'améliorer considérablement la durée du traitement. La plupart de ces routines fonctionnent sur le GS.

## Yvan KOENIG vous propose :

- **Module relogeur**
- **La routine actualisée de Richard Thibert**

```

*****
*          SON , B cont, à incr , F fréq
*          SIR (<,B stfreq, à lfreq, F hfreq >
*
*          ASSEMBLEUR MERLIN
*****
HASADRS = 0      ;1 si table adresses, 0 sinon *
HASDATAS = 1     ;1 si table DATAs, 0 sinon *
EXTERNE = 1      ;1 si commande externe, 0 sinon*
PROTEGE = 0      ;1 si on protège la routine *
INIROUT = 0      ;1 si on saute dans la routine *
*****
DISK   KBD
*****
PUT    RELOPRO
*****
1
2
3  * Module Relogeur destiné à placer une routine *
4  *   entre ProDOS et ses buffers.
5  *
6  * La routine peut être une commande externe,
7  * une simple routine (CALL) ou une routine AMPER
8  *
9  * Prendre garde à la position d'une table
10 * d'adresses et (ou) d'une table de données
11 *
12 * Yvan KOENIG          1e 25-04-86 *
13 *****
14 *
15 *HASADRS = 1      ;1 si table adresses, 0 sinon *
16 *HASDATAS = 1     ;1 si table DATAs, 0 sinon *
17 *EXTERNE = 1      ;1 si commande externe, 0 sinon *
18 *PROTEGE = 0      ;1 si on protège la routine *
19 *INIROUT = 0      ;1 si on saute dans la routine *
20 *
21 *****
22 * On demande à GETBUFR de nous allouer (A) pages *
23 * puis le relogeur copie la routine
24 * dans la zone qui nous est attribuée.
25 * Le relogeur utilise sa propre routine pour
26 * identifier les instructions afin de traiter
27 * correctement les opérateurs spécifiques 65c02
28 * On traitera correctement les appels au MLI
29 * Attention cependant, les tables MLI devront
30 * être placées en zone DATAs et
31 * initialisées par le programme
32 *****
33
34 R_OFFSET = 2      ;&3
35 R_ZBEG = 4        ;&5
36 R_ZEND = 6        ;&7
37 R_A1 = $3C        ;Début du bloc à déplacer
38 R_A2 = $3E        ;Fin du bloc pour MOVE
39 R_A4 = $42        ;Début zone dest. pour MOVE
40 R_INDIR = $FD     ;&FE
41
42 * Adresses ProDOS
43
44 CI_ENTRY = $BE00
45 EXTRNCMD = CI_ENTRY+6
46 GETBUFR = $BEF5
47 MLI = $BF00
48 BITMAP = $BF58
49
50 * Adresses MONITEUR
51
52 R_NXTA4 = $FCB4    ;Avance R_A4 puis tombe en NXTA1
53 R_COUT = $FDED     ;Affiche le caractère (A)
54 R_MOVE = $FE2C     ;Déplace (R_A1 L/H--R_A2 L/H)
55                   ; vers R_A4 L/H
56

```

# SONSIR.S

de Richard Thibert  
en commandes externes ProDOS

```

57 -----
58 ORG $6080
59 -----
60
61 -----
62
63 LDA ELAST-PROGRAM/$100+1
64 JSR GETBUFR ;Libère (A)pages et revient avec
65 ;partie haute adresse dans (A)
66
67 BCC PAGEFND
68 LDY ECANTEND-CANTRELO-1
69 :loop LDA CANTRELO,Y
70 JSR R_COUT
71 DEY
72 BPL :loop
73 JMP CI_ENTRY
74
75 *****
76 CANTRELO HEX BD,87
77 REV "IL N'Y A PAS DE PLACE"
78
79 CANTEND
80 *****
81 OPTABLE DFB %00000100,%00010101,%00000100,%00101010
82 DFB %00000101,%00010100,%00001000,%00101010
83 DFB %00010110,%00010101,%00000100,%00101010
84 DFB %00010101,%00010101,%00001000,%00101010
85 DFB %00000100,%00010101,%00000100,%00101010
86 DFB %00010101,%00010100,%00001000,%00101000
87 DFB %00000100,%00010101,%00000100,%00101010
88 DFB %00010101,%00010101,%00000000,%00101010
89 DFB %00010101,%00010101,%00001000,%00101010
90 DFB %00010101,%00010101,%00000100,%00101010
91 DFB %00000101,%00010101,%00001000,%00101010
92 DFB %00000101,%00010101,%00000100,%00101010
93 DFB %00010101,%00010100,%00001000,%00101000
94 DFB %00000101,%00010101,%00000100,%00101010
95 DFB %00010101,%00010100,%00001000,%00101000
96
97 *****
98
99 PAGEFND STA R_A4+1 ;partie haute adr. zone allouée
100
101 -----
102 DO EXTERNE
103 LDX EXTRNCMD+2 ;sauve l'ancienne adresse pour
104 STX NEXTCMD+2 ;chainer les commandes
105 STA EXTRNCMD+2 ;met en place
106 LDX EXTRNCMD+1 ;la nouvelle adresse
107 STX NEXTCMD+1
108 *

```

```

109 *NEXTCMD JMP #0000 ;On doit trouver ce label
110 *
111 * et cette instruction dans toute
112 * commande externe pour assurer
113 * le chainage des diverses commandes
114 * cf. PRO.TYPC dans Tremplin MICRO n°9
115 *
116 ELSE
117 STA R_INDIR+1 ;Octet haut adresse routine
118 LDA E0 ;relogée
119 STA R_INDIR ; octet bas dito
120 DS 9,#EA ;pour fixer la suite
121
122 FIN
123 -----
124
125 LDA E<PROGRAM ;Début de zone à déplacer
126 STA R_A1
127 STA R_ZBEG
128 LDA E>PROGRAM
129 STA R_A1+1
130 STA R_ZBEG+1
131
132 LDA E<ENDPROG-1 ;Fin de la partie programme
133 STA R_A2
134 LDA E>ENDPROG-1
135 STA R_A2+1
136
137 LDA E<ENDALL-1
138 STA R_ZEND
139 LDA E>ENDALL-1
140 STA R_ZEND+1
141
142 CLD ;Au cas où
143
144 LDA E0 ;Place PROGRAM sur début de page
145 STA R_A4
146
147 -----
148 DO EXTERNE
149 STA EXTRNCMD+1
150 ELSE
151 DS 3,#EA
152 FIN
153 -----
154
155 SEC
156 SBC R_ZBEG ;Calcule offset
157 STA R_OFFSET ; que l'on ajoutera
158 LDA R_A4+1 ; aux adresses
159 SBC R_ZBEG+1 ; devant être 'relogées'
160 STA R_OFFSET+1
161
162 :reloc1 LDY E0
163 LDA (R_A1),Y ;Opcode
164 CMP E#20 ;est-ce JSR ?
165 BNE :notMLI
166 INY
167 LDA (R_A1),Y ;lobyte
168 BNE :notMLI ;ce n'est pas MLI
169 INY
170 LDA (R_A1),Y
171 CMP E>MLI
172 BNE :notMLI
173
174 LDY E0
175 LDX E3-1 ; Pour
176 JSR COPIE ; copier JSR MLI
177 LDA E3-1 ;Trompe la suite pour
178 BNE :opdone ;reloger l'adresse table
179
180 ;Se souvenir qu'après JSR MLI
181 on doit trouver:
182 ; DFB code à exécuter
183 ; DA adresse table paramètres
184 :notMLI LDY E0
185 LDA (R_A1),Y ;Opcode
186 PHA
187 AND E%00000011 ;A= 0 1 2 3 ; Offset
188 TAY ; dans l'octet de la table
189 PLA
190 LSR ; A=OPC/2
191 LSR ; A=OPC/4
192 TAX
193 LDA OPTABLE,X ;contient longueur de 4 opcodes
194 :op1up DEY
195 BMI :opdone ;On a poussé à droite les 2 bits

```

```

196 LSR ;représentant la longueur
197 LSR ;de l'instruction
198 BNE :op1up ;Si A=0, ne branche pas
199 ;Opérande de longueur nulle
200 ; Attention, Y est indéterminé
201 :opdone AND E%00000011 ;Isolé la longueur
202 TAX ; X= 2 1 0
203 LDY E0 ;Lève l'indétermination
204 CPX E3-1
205 BCC :movit1 ;on ne change pas les
206 ;instructions de 1/2 octets
207 INY ; -> Y=1
208 JSR CHANGIT?
209 DEY ; -> Y=0
210 :movit1 ;Ici X=Longueur opérande (0,1,2)
211 JSR COPIE ;Copie l'instruction
212 BCC :reloc1 ;Si (R_A1)<<(R_A2)
213
214 -----
215 :mov2? DO HASADRS
216 BCC :mov3? ;Ne branche pas si table Adresses
217
218 ELSE
219 BCS :mov3? ;Branche si pas de table Adresses
220
221 FIN
222 -----
223 LDA E<ENDADRS-1 ;Fin de table d'adresses
224 STA R_A2
225 LDA E>ENDADRS-1
226 STA R_A2+1
227 :reloc2 ;Ici Y=0
228 JSR CHANGIT?
229
230 :movit2 LDX E2-1 ;2 octets par adresse
231 JSR COPIE
232 BCC :reloc2 ;si l'on n'a pas fini la table
233
234 -----
235 :mov3? DO HASDATAS
236 BCC :exit ;Ne branche pas si table DATAS
237 ELSE
238 BCS :exit ;Branche si pas de table DATAS
239 FIN
240 -----
241
242 LDA R_ZEND ;Fin de table de valeurs
243 STA R_A2
244 LDA R_ZEND+1
245 STA R_A2+1
246
247 JSR R_MOVE
248 :exit JSR PROTECT
249
250 -----
251 DO INIROUT
252 NOP
253 ELSE
254 RTS
255 FIN
256 -----
257
258 JMP (R_INDIR) ;Exécute si INIROUT = 1
259
260 =====
261 *
262 * C'est relogé
263 *
264 =====
265
266 CHANGIT?
267 LDA R_ZEND
268 CMP (R_A1),Y
269 INY
270 LDA R_ZEND+1
271 SBC (R_A1),Y
272 BCC :end ;Pas de changement
273
274 DEY
275 LDA R_ZBEG
276 CMP (R_A1),Y
277 INY
278 LDA R_ZBEG+1

```

```

279     SBC  (R_A1),Y
280     BCS  :end          ;Pas de changement
281                               ;Ici C=0
282     DEY
283     LDA  (R_A1),Y
284     ADC  R_OFFSET      ;Ajuste
285     STA  (R_A1),Y      ; l'adresse
286     INY
287     LDA  (R_A1),Y
288     ADC  R_OFFSET+1
289     STA  (R_A1),Y
290
291 :end   DEY
292       RTS
293
294 *****
295
296 COPIE  LDA  (R_A1),Y    ;Copie X+1 octets
297       STA  (R_A4),Y    ;à leur destination
298       JSR  R_NXTA4
299       DEX
300       BPL  COPIE
301       RTS
302
303 *****
304
305 PROTECT DO  PROTEGE
306         NDP                ;On protégera
307         ELSE
308         RTS                ;On ne protégera pas
309         FIN
310
311     LDA  R_ZBEG+1
312     CLC
313     ADC  R_OFFSET
314     TAX                ;(Haut) plus basse page utilisée
315     CLC
316     ADC  $LAST-PROGRAM/$100+1 ;Nombre de pages
317     STA  R_A4+1        ;(Haut) page suivant la routine
318 :loop  TXA
319       PHA
320       LSR
321       LSR
322       LSR
323       TAY
324       TXA
325       AND  $7
326       TAX
327       LDA  $0
328       SEC
329 :L     ROR
330       DEX
331       BPL  :L
332       ORA  BITMAP,Y
333       STA  BITMAP,Y
334       PLA
335       TAX
336       INX
337       CPX  R_A4+1
338       BCC  :loop
339       RTS
340
341     ERR  *-1/$6200
342     DS   $6200-*      ;Remplissage
343
344 -----
345 PROGRAM PUT  SONSIR    ;Controler début de page
346
347 *ORG $6200 ;pour mise au point
348 *****
349 *
350 *          SONSIR.S *
351 *
352 *  Routines de R. THIBERT *
353 *  parues dans Tremplin MICRO 5 *
354 *  réécrites en commandes *
355 *  externes PRODOS *
356 *
357 *  Yvan KOENIG      14/02/87 *
358 *****
359
360 CONT  =  $3C
361 STFREQ =  CONT
362 INCR  =  $3D
363 LFREQ =  INCR
364 FREQ  =  $3E
365 HFREQ =  FREQ

```

```

20 KMD  =  $3F
21 P    =  KMD
22
23 CHRGOT =  $B7
24
25 *****
26
27 DOSETRY JMP  PARSE      ; Quelques instructions
28 NEXTCMD
29 DOSEXIT  JMP  XRETURN   ; pour ceux qui souhaitent
30 AMPEXIT  JMP  SURE_RTS  ; intégrer cette commande
31                               ; à ProCMD
32 AMPENTRY BNE  AMPEXIT   ; Avec ProCMD '&'
33         LDY  $0         ; affiche la liste
34 :1       LDA  CMDNAME,Y ; des commandes
35         JSR  COUT       ; mises en place
36         INY           ; avec leur syntaxe
37         CPY  $AMPEND+1-CMDNAME
38         BCC  :1
39         JSR  CHRGOT
40         BEQ  AMPEXIT
41
42 IDBYTE  NOP            ;Opcode 1 octet pour mon relogeur
43
44 PARSE   LDY  $(CMDNAME-1
45         LDA  IN+2
46         AND  $DF        ; Min->MAJ
47         EOR  $N"
48         STA  KMD        ; $00=SON, $1C=SIR
49         BEQ  :1         ; Branche si SON
50         LDY  $(CMDNAME2-1
51 :1      STY  :3+1
52         LDX  $CMDEND-CMDNAME-1
53         STX  XLEN
54         INX
55 :2      LDA  IN-1,X
56         AND  $DF        ; min->MAJ
57 :3      EOR  CMDNAME-1,X ; pointe SON ou SIR
58         SEC
59         BNE  DOSEXIT
60         DEX
61         BNE  :2
62         STX  XCNUM      ; X=0
63         LDA  $00010000
64         STA  PBITS      ; Force analyse des paramètres
65         LDA  $01001010 ; Autorise B, à, F
66         STA  PBITS+1
67         LDA  $TRUECMD
68         STA  XTRNADDR
69         LDA  HIBIT
70 HIBIT  =  *-1          ; Donne adresse
71         STA  XTRNADDR+1 ; d'entrée à ProDOS
72         CLC
73         RTS
74
75 *****
76
77 TRUECMD PHP
78         SEI            ;Interdit les interruptions
79         LDX  $A0       ;STFREQ par défaut
80         LDA  FBITS+1
81         AND  $01000000 ;A-t-on spécifié B (CONT/STFREQ)
82         BEQ  :B        ;NON -> valeur par défaut
83         LDX  VBYTE
84 :B     STX  CONT
85
86         LDX  $50       ;LFREQ par défaut
87         LDA  FBITS+1
88         AND  $00001000 ;A-t-on spécifié à (INCR/LFREQ)
89         BEQ  :L        ;NON
90         LDX  VLINE
91 :L     STX  INCR
92
93         LDX  $A0       ;HFREQ par défaut
94         LDA  FBITS+1
95         AND  $00000010 ;A-t-on spécifié F (FREQ/HFREQ)
96         BEQ  :F        ;NON
97         LDX  VFELD
98 :F     STX  FREQ
99
100        LDA  KMD
101        BNE  SIRENE
102
103 SOND1  LDA  $0
104        STA  p

```

```

105 :1      LDX  FREQ
106 :2      LDA  P
107          CLC
108          ADC  INCR
109          STA  P
110          TAY
111 :3      DEY
112          BNE  :3
113          LDA  SPK      ;CLIC
114          TXA
115          TAY
116 :4      DEY
117          BNE  :4
118          LDA  SPK      ;CLIC
119          DEX
120          BNE  :2
121          DEC  CONT
122          BNE  :1
123
124 EXIT    STA  STROBE
125          PLP          ;rétablit état interruptions
126          CLC          ;pour sortie correcte
127          RTS
128
129 -----
130
131 SIRENE  LDA  STROBE    ;Clavier à zéro
132          LDX  STFREQ
133 :1      TXA
134 :2      DEX
135          NOP
136          BNE  :2
137          TAX
138          LDA  SPK      ;CLIC
139          LDA  KBD      ;A-t-on frappé une touche ?
140          BMI  EXIT    ;OUI -> c'est fini
141          BCC  :3      ;Monter ou Descendre selon C
142          DEX          ;tout schuss
143          CPX  LFREQ    ;positionne C
144          JMP  :1
145
146 :3      INX          ;montons
147          CPX  HFREQ    ;positionne C
148          JMP  :1
149
150 *****
151 ENDPROG = *
152 *****
153 ENDADRS = *
154 *****
155
156 *****
157 * Dans ProcCMD, taper & affiche la liste *
158 * des commandes actives et leur syntaxe *
159 *****
160
161 * CMDNAME et CMDNAME2 doivent être dans une meme page
162
163 CMDNAME  ASC  "SON"
164 CMDEND   ASC  " ou "
165
166 CMDNAME2 ASC  "SIR"
167          ASC  " ,B E, à E, F E"
168 AMPEND   HEX  8D

```

```

169 LAST    =    *-1
170 ENDALL   =    *
171 *****
172
173 IN       =    $200
174
175 XTRNADDR =    $BE50
176 XLEN     =    $BE52
177 XCNUM    =    $BE53
178 PBITS    =    $BE54      ;&BE55
179 FBITS    =    $BE56      ;&BE57
180 VBYTE    =    $BE5A      ; paramètre B
181 VFELD    =    $BE63      ; paramètre F
182 VLINE    =    $BE68      ; paramètre à
183
184 XRETURN  =    $BE9E
185
186 KBD      =    $C000
187 STROBE   =    $C010
188 SPK      =    $C030
189
190 COUT     =    $FDED
191 SURE_RTS =    $FFCB
192 *****
193          DO    DISK
194          SAV  SONSIR
195          FIN
196 *****

```

Faire - PRO.FP  
 - SONSIR  
 dans le STARTUP

### SONSIR.DEMO

```

10 TEXT : HOME : PRINT "Tapez une touche po          154C
    ur arrêter les sirènes"
20 D$ = CHR$(4)          83A4
30 LIST 40              5C20
40 PRINT D$"SIR"        2D54
50 PRINT : PRINT "INDICATIF"          0A7D
60 A = 103:X = 50:Y = 130: GOSUB 500   74E2
70 PRINT : PRINT "MUSIQUE"            8D1B
80 A = 16:X = 120:Y = 1: GOSUB 500    FF80
90 PRINT : PRINT "PROMPT"             05D4
100 A = 1:X = 255:Y = 35: GOSUB 500   388A
110 PRINT : PRINT "SIRENE"            E4BB
120 Y = 255: GOSUB 500                2E44
130 PRINT : PRINT "TUUT"              4244
140 Y = 1: GOSUB 500                  0FD9
150 PRINT : PRINT "SLURP"              E188
160 A = 128:X = 6:Y = 254: GOSUB 500  3FC1
170 LIST 180                        2B55
180 PRINT D$"SIR,B32,F48,à32"         FFD6
190 PRINT : PRINT "AUX ABRIS !!!"     61F4
200 LIST 210                          174F
210 PRINT D$"SON"                      E956
220 LIST 230                            2351
230 PRINT D$"SIR,B128,à0,F255"       9C07
240 END                                0180
500 PRINT D$"SON,B"A",F"X",à"Y: GET X$: PRIN
    T : RETURN                        D56F

```



## SONSIR en commandes externes ProDOS

```
6080: A9 01 20 F5 BE 90 65 A0 16 B9 95 60 20 ED FD 88 5568
6090: 10 F7 4C 00 BE 8D 87 C5 C3 C1 CC D0 A0 C5 C4 A0 5DD3
60A0: D3 C1 D0 A0 C1 A0 D9 A7 CE A0 CC C9 04 15 04 2A D02F
60B0: 05 14 08 2A 16 15 04 2A 15 15 08 2A 04 15 04 2A 3247
60C0: 15 14 08 28 04 15 04 2A 15 15 08 2A 15 15 00 2A 0650
60D0: 15 15 08 2A 15 15 04 2A 05 15 08 2A 05 15 04 2A 1548
60E0: 15 14 08 28 05 15 04 2A 15 14 08 28 85 43 AE 08 0578
60F0: BE BE 05 62 8D 08 BE AE 07 BE 8E 04 62 A9 00 85 3998
6100: 3C 85 04 A9 62 85 3D 85 05 A9 D7 85 3E A9 62 95 94EF
6110: 3F A9 F1 85 06 A9 62 85 07 D8 A9 00 85 42 8D 07 C2D7
6120: BE 38 E5 04 85 02 A5 43 E5 05 85 03 A0 00 B1 3C E44D
6130: C9 20 D0 17 C8 B1 3C D0 12 C8 B1 3C C9 BF D0 0B 0E7F
6140: A0 00 A2 02 20 C3 61 A9 02 D0 16 A0 00 B1 3C 48 B6EE
6150: 29 03 A8 68 4A 4A AA BD AC 60 88 30 04 4A 4A D0 E863
6160: F9 29 03 AA A0 00 E0 02 90 05 C8 20 9C 61 88 20 D373
6170: C3 61 90 88 B0 12 A9 D7 85 3E A9 62 85 3F 20 9C 4DFC
6180: 61 A2 01 20 C3 61 90 F6 90 08 A5 06 85 3E A5 07 4C83
6190: 85 3F 20 2C FE 20 CE 61 60 6C FD 00 A5 06 D1 3C 91DE
61A0: C8 A5 07 F1 3C 90 1A 88 A5 04 D1 3C C8 A5 05 F1 F6EC
61B0: 3C B0 0E 88 B1 3C 65 02 91 3C C8 B1 3C 65 03 91 5951
```

```
61C0: 3C 88 60 B1 3C 91 42 20 B4 FC CA 10 F6 60 60 A5 FCE9
61D0: 05 18 65 02 AA 18 69 01 85 43 8A 48 4A 4A 4A A8 29D0
61E0: 8A 29 07 AA A9 00 38 6A CA 10 FC 19 58 BF 99 58 A3A6
61F0: BF 68 AA E8 E4 43 90 E2 60 00 00 00 00 00 00 00 A8B2
6200: 4C 1E 62 4C 9E BE 4C C8 FF D0 FB A0 00 B9 D8 62 9DE8
6210: 20 ED FD C8 C0 1A 90 F5 20 B7 00 F0 E9 EA A0 07 4842
6220: AD 02 02 29 DF 49 CE 85 3F F0 02 A0 DE 8C 3C 62 082E
6230: A2 02 8E 52 BE E8 BD FF 01 29 DF 5D D7 62 38 D0 CF8D
6240: C2 CA D0 F2 8E 53 BE A9 10 8D 54 BE A9 4A 8D 55 721A
6250: BE A9 5E 8D 50 BE AD 58 62 8D 51 BE 18 60 08 78 C758
6260: A2 A0 AD 57 BE 29 40 F0 03 AE 5A BE 86 3C A2 50 C8DA
6270: AD 57 BE 29 08 F0 03 AE 68 BE 86 3D A2 A0 AD 57 15C3
6280: BE 29 02 F0 03 AE 63 BE 86 3E A5 3F D0 29 A9 00 12F5
6290: 85 3F A6 3E A5 3F 18 65 3D 85 3F A8 88 D0 FD AD 86B4
62A0: 30 C0 8A A8 88 D0 FD AD 30 C0 CA D0 E7 C6 3C D0 7E67
62B0: E1 8D 10 C0 28 18 60 AD 10 C0 A6 3C 8A CA EA D0 1048
62C0: FC AA AD 30 C0 AD 00 C0 30 E7 90 06 CA E4 3D 4C F994
62D0: BC 62 E8 E4 3E 4C BC 62 D3 CF CE A0 EF F5 A0 D3 FDF9
62E0: C9 D2 A0 AC C2 A0 A3 AC A0 C0 A0 A3 AC A0 C6 A0 CFED
62F0: A3 8D DC30
```

BSAVE SONSIR.A\$6080.L\$272

# PRO.FP

Nous avons publié les codes de PRO.FP dans Tremplin Micro n°11, mais pas le source.

Vous savez que PRO.FP remet en place tous les pointeurs et — ce qui est nouveau — remet les buffers dans leur position normale.

C'est la réplique, sous ProDOS, de la commande FP du DOS 3.3.

Avec PRO.FP, on évite que des initialisations successives de certaines routines ne réduisent finalement la mémoire disponible à zéro.

(suite page 66)

```
1
2 *****
3 *
4 * PRO.FP *
5 *
6 * Yvan KOENIG *
7 *
8 *****
9
10 TXTTAB = $67
11 SOFTEV = $3F2
12 AMPERV = $3F5
13
14 CI_ENTRY = $BE00
15 DOSCMD = $BE03
16 EXTRNCMD = $BE06
17 OUTVECT0 = $BE10
18 OUTVECT1 = $BE12
19 OUTVECT3 = $BE16
20 INJECT0 = $BE20
21 INJECT1 = $BE22
22 INJECT3 = $BE26
23 XRETURN = $BE9E
24 FREEBUFR = $BEF8
25 MLI = $BF00
26 BITMAP = $BF58
27 INIT = $FB2F
28 HOME = $FC58
29 KEYIN = $FD1B
30 COUT = $FDED
31 COUT1 = $FDF0
```

Révisé //GS  
1e 10/03/87

```

32  IDROUTINE =  $FE1F
33  SETNORM  =   $FE84
34
35          ORG  $300
36
300: 20 F8 BE 37          JSR  FREEBUFR  ; Libère les pages utilisées
303: A9 9E   38          LDA  £<XRETURN ; Supprime les commandes
                                   externes
305: 8D 07 BE 39          STA  EXTRNCMD+1
308: A9 BE   40          LDA  £>XRETURN
30A: 8D 08 BE 41          STA  EXTRNCMD+2
30D: A9 03   42          LDA  £<DOSCMD
30F: 8D F6 03 43          STA  AMPERV+1  ; Revectorise Ampersand
312: A9 BE   44          LDA  £>DOSCMD
314: 8D F7 03 45          STA  AMPERV+2
317: A9 08   46          LDA  £>$801  ; Rétablit début normal
319: 85 68   47          STA  TXTTAB+1 ; du BASIC
31B: A2 01   48          LDX  £<$801
31D: 86 67   49          STX  TXTTAB
31F: CA     50          DEX
320: 8E 00 08 51          STX  $800
323: 8A     52          TXA
324: A2 12   53          LDX  £18  ; Corrige la Bitmap
326: 9D 58 BF 54  §L    STA  BITMAP,X
329: CA     55          DEX
32A: D0 FA   56          BNE  §L
32C: A9 3F   57          LDA  £$3F
32E: 8D 6B BF 58          STA  BITMAP+19
331: A9 CF   59          LDA  £$CF
333: 8D 58 BF 60          STA  BITMAP
336: A9 F0   61          LDA  £<COUT1  ; corrige les vecteurs I/O
338: 8D 10 BE 62          STA  OUTVECT0 ; numéros 0,1 et 3
33B: A9 FD   63          LDA  £>COUT1
33D: 8D 11 BE 64          STA  OUTVECT0+1
340: 8D 21 BE 65          STA  INVECT0+1
343: A9 1B   66          LDA  £<KEYIN
345: 8D 20 BE 67          STA  INVECT0
348: A9 00   68          LDA  £0
34A: 8D 12 BE 69          STA  OUTVECT1
34D: 8D 16 BE 70          STA  OUTVECT3
350: 8D 22 BE 71          STA  INVECT1
353: 8D 26 BE 72          STA  INVECT3
356: A9 C1   73          LDA  £>$C100
358: 8D 13 BE 74          STA  OUTVECT1+1
35B: 8D 23 BE 75          STA  INVECT1+1
35E: A9 C3   76          LDA  £>$C300
360: 8D 17 BE 77          STA  OUTVECT3+1
363: 8D 27 BE 78          STA  INVECT3+1
366: A9 95   79          LDA  £21+$80  ; Ctrl U déconnecte 80 cols
368: 20 ED FD 80          JSR  COUT
36B: 20 84 FE 81          JSR  SETNORM
36E: 20 2F FB 82          JSR  INIT

```

```

371: 20 58 FC 83      JSR  HOME
374: A9 00 84      LDA  £<CI_ENTRY ; Revectorise RESET
376: 8D F2 03 85      STA  SOFTEV
379: A9 BE 86      LDA  £>CI_ENTRY
37B: 8D F3 03 87      STA  SOFTEV+1
37E: 49 A5 88      EOR  £$A5
380: 8D F4 03 89      STA  SOFTEV+2
383: AD B3 FB 90      LDA  $FBB3
386: 30 0E 91      BMI  FLUSHIRQ ; un II+
388: AD C0 FB 92      LDA  $FBC0
38B: F0 09 93      BEQ  FLUSHIRQ ; un //c
38D: C9 E0 94      CMP  £$E0
38F: D0 05 95      BNE  FLUSHIRQ ; un //e vieilles Roms
          96      ; Ici C=1
391: 20 1F FE 97      JSR  IDROUTINE ; identification OFFICIELLE
394: 90 0D 98      BCC  EXIT ; C'est un //gs
396: A2 01 99      FLUSHIRQ LDX  £1
398: 8E AD 03 100    $L      STX  PRIORITY ; déconnecte les interruptions
39B: 20 A4 03 101    JSR  DEALLOC
39E: E8 102      INX
39F: E0 05 103    CPX  £4+1
3A1: 90 F5 104    BCC  $L
3A3: 60 105      EXIT  RTS
          106
3A4: 78 107      DEALLOC SEI
3A5: 20 00 BF 108    JSR  MLI
3A8: 41 109      DFB  $41
3A9: AC 03 110    DA   PARM5
3AB: 60 111      RTS
          112
3AC: 01 113      PARM5  DFB  1
3AD: 01 114      PRIORITY DFB  1

```

-End assembly, 174 bytes, Errors: 0

## PRO.FP

BSAVE PRO.FP,A\$300.L174

```

0300: 20 F8 BE A9 9E 8D 07 BE A9 BE 8D 08 BE A9 03 8D CF62
0310: F6 03 A9 BE 8D F7 03 A9 08 85 68 A2 01 86 67 CA 68DF
0320: 8E 00 08 8A A2 12 9D 58 BF CA D0 FA A9 3F 8D 6B 5BFC
0330: BF A9 CF 8D 58 BF A9 F0 8D 10 BE A9 FD 8D 11 BE 11D1
0340: 8D 21 BE A9 1B 8D 20 BE A9 00 8D 12 BE 8D 16 BE 5802
0350: 8D 22 BE 8D 26 BE A9 C1 8D 13 BE 8D 23 BE A9 C3 1980
0360: 8D 17 BE 8D 27 BE A9 95 20 ED FD 20 84 FE 20 2F 8B0D
0370: FB 20 58 FC A9 00 8D F2 03 A9 BE 8D F3 03 49 A5 E172
0380: 8D F4 03 AD B3 FB 30 0E AD C0 FB F0 09 C9 E0 D0 23F7
0390: 05 20 1F FE 90 0D A2 01 8E AD 03 20 A4 03 E8 E0 934F
03A0: 05 90 F5 60 78 20 00 BF 41 AC 03 60 01 01 0293

```

### Notre ami Yvan KOENIG

répond volontiers aux questions des lectrices et lecteurs de *TREMLIN MICRO*, mais ceux-ci doivent respecter trois règles impératives :

- Joindre une enveloppe timbrée à leur adresse.
- Ne poser qu'une question par lettre.
- Nous envoyer un disque si cette question concerne une routine.

Nous signalons à nos nouveaux lecteurs que chaque disquette *TREMLIN MICRO* contient tous les programmes parus dans le numéro correspondant de la revue (une face DOS 3.3 et une face ProDOS).

# FENÊTRES HGR

**J**ACQUES FOURNEAU vous propose des fenêtres en mode graphique. Etudiez son programme source, puis offrez-vous une démonstration avec les quelques lignes ci-après. Notez que les variables **C1** et **C2** contiendront respectivement les limites gauche et droite de la fenêtre (0 à 279), tandis que **L1** et **L2** seront réservées à la ligne du haut et à celle du bas (0 à 191).

## FEN.HGR.BAS

Sous DOS 3.3,  
fixez HIMEM  
à 38144 (ligne 10)  
et supprimez  
la ligne 20.

Pour obtenir  
des fenêtres  
en HGR,  
remplacez HGR2  
(ligne 40)  
par HGR.  
La routine  
en langage  
machine se  
chargera du reste.

```

10 TEXT : HOME : PRINT CHR$(21): HIMEM: 38400      9CC5
15 PRINT CHR$(4)"BLOAD FEN.HGR"                    C0D4
20 HIMEM: 38144                                     1CA7
25 POKE 1013,76: POKE 1014,0: POKE 1015,149        5CB0
30 HOME                                             2F97
35 INPUT C1: INPUT C2: INPUT L1: INPUT L2          1BA2
40 HGR2 : HCOLOR= 3: FOR I = 0 TO 279: HPLLOT I,0 TO  2CF5
   I,191: NEXT                                     8148
45 CALL - 198: GET A$: PRINT                        8C17
50 & C1,C2,L1,L2                                    19B1
55 GOSUB 85: TEXT : HOME                            20A9
60 VTAB 22: PRINT "(E)NCORE (T)ERMINE (M)ENU DE DIS  6792
   QUETTE ";: GOSUB 85                             868F
65 IF A$ = "E" THEN 30                             A76D
70 HOME : IF A$ = "T" THEN END                      8A55
75 IF A$ < > "M" THEN 60                            BC02
80 PRINT CHR$(4)"RUN MENU,D1"
85 GET A$: PRINT : RETURN

```

SUITE PAGE 68

## ATTENTION !

A la suite d'un réorganisation des services rédactionnels de *TREMLIN MICRO*, il ne nous est plus possible de fournir des renseignements *techniques* par téléphone.

Vous pouvez par contre nous écrire, mais n'oubliez pas de joindre une enveloppe timbrée pour la réponse. MERCI !

Vous trouverez  
la liste des codes  
de ce programme  
page 71.

	0	*	EFFACER PARTIE ECRAN GRAPHIQUE *	
	1	*		
	2		ORG \$9500	
	3	ADLMEM8	EQU \$06	;MEMOIRE ADRESSE POUR LIGNE TEXTE
	4	ADHMEM8	EQU \$07	
	5	ADLMEM3	EQU \$08	;MEMOIRE ADRESSE TIERS PAGE
	6	ADHMEM3	EQU \$09	
	7	HPAG	EQU \$E6	;SELECTEUR PAGE HGR (1=\$20 2=\$40)
	8	COUNT	EQU \$D7	;COMPTEUR LIGNES (0 A 191)
	9	XDEB	EQU \$EB	;POSITION DANS OCTET GAUCHE (0A6)
	10	XFIN	EQU \$EC	;POSITION DANS OCTET DROITE (0A6)
	11	ADL	EQU \$ED	;ADR. DE TRAVAIL DANS PAGE HGR
	12	ADH	EQU \$EE	
	13	QUOTIENT	EQU \$EF	;CASE DE TRAVAIL POUR DIVISION/7
	14	BYTE1	EQU \$F9	;CASE POUR COMPARAISON
	15	C1	EQU \$FA	;NO OCTETGAUCHE (0 A 39)
	16	C2	EQU \$FC	;IDEM DROITE
	17	L1	EQU \$FE	;HAUT FENETRE (0 A 191)
	18	L2	EQU \$FF	;BAS FENETRE
	19	*		
9500:	20	7B DD	DEBUT JSR \$DD7B	;EVALUATION FORMULE APRES &
9503:	21	52 E7	JSR \$E752	;TRANSFORME VAL. SUR 2 OCT. (Y-A)
9506:	22	84 FA	STY C1	
9508:	23	85 FB	STA C1+1	
950A:	24	20 BE DE	JSR \$DEBE	;EVALUATION VIRGULE
950D:	25	20 7B DD	JSR \$DD7B	
9510:	26	20 52 E7	JSR \$E752	
9513:	27	84 FC	STY C2	
9515:	28	85 FD	STA C2+1	
9517:	29	20 BE DE	JSR \$DEBE	
951A:	30	20 F8 E6	JSR \$E6F8	;VALEUR SUR 1 OCTET (DANS X)
951D:	31	86 FE	STX L1	
951F:	32	20 BE DE	JSR \$DEBE	
9522:	33	20 F8 E6	JSR \$E6F8	
9525:	34	86 FF	STX L2	
	35	*		
9527:	36	A5 FA	CALCULC1 LDA C1	;PREMIERE VALEUR A DIVISER
9529:	37	85 EF	STA QUOTIENT	;PAR 7 POUR CONNAITRE
952B:	38	A5 FB	LDA C1+1	;LE NO DE L'OCTET
952D:	39	20 DC 95	JSR DIVIS7	
9530:	40	85 EB	STA XDEB	;RESTE RECU. POUR POS.DANS OCTET
9532:	41	A5 EF	LDA QUOTIENT	
9534:	42	85 FA	STA C1	
9536:	43	A5 FC	CALCULC2 LDA C2	;MEME TRAVAIL POUR C2.
9538:	44	85 EF	STA QUOTIENT	
953A:	45	A5 FD	LDA C2+1	
953C:	46	20 DC 95	JSR DIVIS7	
953F:	47	85 EC	STA XFIN	
9541:	48	A5 EF	LDA QUOTIENT	
9543:	49	85 FC	STA C2	
	50	*		
9545:	51	A5 E6	INIT LDA HPAG	;CHARGER ADRESSE (\$20 OU \$40)

9547:	85 EE	52	STA	ADH	;DE DEPART SUIVANT PAGE
9549:	85 09	53	STA	ADHMEM3	;LA SAUVER AVANT SAUTS
954B:	85 07	54	STA	ADHMEM8	;IDEM
954D:	A9 00	55	LDA	£\$00	
954F:	85 D7	56	STA	COUNT	;COMPTEUR A ZERO
9551:	85 ED	57	STA	ADL	
9553:	85 08	58	STA	ADLMEM3	
9555:	85 06	59	STA	ADLMEM8	
9557:	AA	60	TAX		
		61	*		
9558:	A5 D7	62	MAIN	LDA	COUNT
955A:	C5 FE	63		CMP	L1 ;LIGNE DEPART?
955C:	90 25	64		BCC	SUITE ;SI NON ON CHANGE COMPTEUR & ADR.
		65	*		
955E:	A4 EB	66	CHARGE	LDY	XDEB ;RECHERCHE DANS LA
9560:	B9 CE 95	67		LDA	OCTETGA,Y ;TABLE DE L'OCTET DE COMPARAISON
9563:	85 F9	68		STA	BYTE1 ;ET MEMORISATION
9565:	A4 FA	69		LDY	C1
9567:	B1 ED	70		LDA	(ADL),Y ;CHARG. DE L'ADR. DU 1er OCTET
9569:	25 F9	71		AND	BYTE1 ;DONT LA VALEUR EST MODIFIEE
956B:	91 ED	72	BOUCLE	STA	(ADL),Y
956D:	C8	73		INY	
956E:	A9 00	74		LDA	£\$00 ;MODIF. DES OCTETS INTERMEDIAIRES
9570:	C4 FC	75		CPY	C2 ;COLONNE FIN ?
9572:	90 F7	76		BCC	BOUCLE ;NON, ON CONTINUE LA BOUCLE
9574:	A4 EC	77		LDY	XFIN ;OUI, ON RECHERCHE
9576:	B9 D5 95	78		LDA	OCTETDR,Y ;DANS LA TABLE L'OCTET DE COMP.
9579:	85 F9	79		STA	BYTE1 ;QUE L'ON MEMORISE
957B:	A4 FC	80		LDY	C2
957D:	B1 ED	81		LDA	(ADL),Y ;CHARG. DE L'ADR. DU DERN. OCTET
957F:	25 F9	82		AND	BYTE1 ;DONT LA VALEUR EST MODIFIEE
9581:	91 ED	83		STA	(ADL),Y
		84	*		
9583:	E6 D7	85	SUITE	INC	COUNT ;LIGNE SUIVANTE
9585:	A4 D7	86		LDY	COUNT
9587:	C4 FF	87		CPY	L2 ;DERNIERE LIGNE?
9589:	F0 02	88		BEQ	NEXTLINE ;OUI, IL FAUT LA FAIRE
958B:	B0 40	89		BCS	FIN ;>L2, C'EST FINI
958D:	A5 EE	90	NEXTLINE	LDA	ADH
958F:	18	91		CLC	
9590:	69 04	92		ADC	£\$04 ;SAUT DE 1024
9592:	85 EE	93		STA	ADH ;POUR TROUVER LIGNE SUIVANTE
9594:	E8	94		INX	
9595:	E0 08	95		CPX	£\$08 ;8 LIGNES GRAP. = UNE LIGNE TEXTE
9597:	90 BF	96		BCC	MAIN ;SI<8 ON RETOURNE AU PROG. PRINC.
9599:	A2 00	97		LDX	£\$00 ;SI = 8, ON REMET X A ZERO
		98	*		
959B:	C0 40	99		CPY	£\$40 ;FIN 1er 1/3 DE PAGE (=64 LGNS)?
959D:	F0 1A	100		BEQ	PLUS ;OUI, ON VA MODIFIER ADRESSE
		101	*		
959F:	C0 80	102		CPY	£\$80 ;FIN 2e TIERS (=128 LIGNES) ?
95A1:	F0 16	103		BEQ	PLUS ;OUI, ON VA MODIFIER ADRESSE
		104	*		
95A3:	A5 07	105		LDA	ADHMEM8 ;NON, ON RECUP. L'ADR. MEMORISEE

(Suite page 70)

```

95A5: 85 EE      106      STA  ADH
95A7: A5 06      107      LDA  ADLMEM8
95A9: 18          108      CLC
95AA: 69 80      109      ADC  £$80      ;AJOUT. 128 POUR PASSAGE LIGNE TXT
95AC: 85 06      110      STA  ADLMEM8
95AE: 85 ED      111      STA  ADL
95B0: 90 A6      112      BCC  MAIN
95B2: E6 07      113      INC  ADHMEM8   ;S'IL Y A EU CARRY
95B4: E6 EE      114      INC  ADH       ;ON INCREMENTE L'ADRESSE
95B6: 4C 58 95   115      JMP  MAIN
          116      *
95B9: A5 08      117  PLUS  LDA  ADLMEM3   ;ON RECUPERE L'ADRESSE
95BB: 18          118      CLC           ;AJOUTER 40 POUR PASSER
95BC: 69 28      119      ADC  £$28     ;VERS TIERS ECRAN SUIVANT
95BE: 85 ED      120      STA  ADL
95C0: 85 06      121      STA  ADLMEM8
95C2: 85 08      122      STA  ADLMEM3
95C4: A5 09      123      LDA  ADHMEM3
95C6: 85 EE      124      STA  ADH
95C8: 85 07      125      STA  ADHMEM8
95CA: 18          126      CLC
95CB: 90 8B      127      BCC  MAIN
95CD: 60          128  FIN  RTS
          129      *
          130      * TABLE DE VALEURS POUR MASQUE DANS
          131      * OCTETS EXTREMES DE LA LIGNE

95CE: 00 01 03
          07 0F 1F
          3F      132  OCTETGA  DFB  $00,$01,$03,$07,$0F,$1F,$3F
95D5: FE FC F8
          F0 E0 C0
          80      133  OCTETDR  DFB  $FE,$FC,$F8,$F0,$E0,$C0,$80
          134      *
95DC: A2 08      135  DIVIS7  LDX  £$08      ;HUIT BITS
95DE: 06 EF      136  DIV      ASL  QUOTIENT
95E0: 2A          137      ROL           ;A CONTIENDRA LE RESTE DE LA DIV.
95E1: C9 07      138      CMP  £$07
95E3: 90 04      139      BCC  DIVSUITE
95E5: E9 07      140      SBC  £$07
95E7: E6 EF      141      INC  QUOTIENT
95E9: CA          142  DIVSUITE DEX
95EA: D0 F2      143      BNE  DIV
95EC: 60          144      RTS

```

#### Table des symboles ordre alphanetique

-ADH.....\$00EE	-ADHMEM3...\$0009	-ADHMEM8...\$0007	-ADL.....\$00ED
-ADLMEM3...\$0008	-ADLMEM8...\$0006	-BOUCLE...\$956B	-BYTE1.....\$00F9
-C1.....\$00FA	-C2.....\$00FC	? -CALCULC1...\$9527	? -CALCULC2...\$9536
? -CHARGE...\$955E	-COUNT....\$00D7	? -DEBUT....\$9500	-DIV.....\$95DE
-DIVIS7...\$95DC	-DIVSUITE..\$95E9	-FIN.....\$95CD	-HPAG.....\$00E6
? -INIT.....\$9545	-L1.....\$00FE	-L2.....\$00FF	-MAIN.....\$9558
-NEXTLINE..\$958D	-OCTETDR...\$95D5	-OCTETGA...\$95CE	-PLUS.....\$95B9
-QUOTIENT..\$00EF	-SUITE.....\$9583	-XDEB.....\$00EB	-XFIN.....\$00EC

## **FEN.HGR**

BSAVE FEN.HGR,A\$9500,L\$ED

(codes)

Avec la disquette  
**SIGNATURE** de  
**TREMLIN MICRO**,  
recopier une routine  
en langage machine  
ne pose plus de  
problème, grâce à un  
contrôle facile de  
l'exactitude des codes.

9500:	20 7B DD 20 52 E7 84 FA 85 FB 20 BE DE 20 7B DD	8E03
9510:	20 52 E7 84 FC 85 FD 20 BE DE 20 F8 E6 86 FE 20	65B9
9520:	BE DE 20 F8 E6 86 FF A5 FA 85 EF A5 FB 20 DC 95	F963
9530:	85 EB A5 EF 85 FA A5 FC 85 EF A5 FD 20 DC 95 85	6550
9540:	EC A5 EF 85 FC A5 E6 85 EE 85 09 85 07 A9 00 85	4347
9550:	D7 85 ED 85 08 85 06 AA A5 D7 C5 FE 90 25 A4 EB	FA8E
9560:	B9 CE 95 85 F9 A4 FA B1 ED 25 F9 91 ED C8 A9 00	FFE3
9570:	C4 FC 90 F7 A4 EC B9 D5 95 85 F9 A4 FC B1 ED 25	C6DB
9580:	F9 91 ED E6 D7 A4 D7 C4 FF F0 02 B0 40 A5 EE 18	78FF
9590:	69 04 85 EE E8 E0 08 90 BF A2 00 C0 40 F0 1A C0	706B
95A0:	80 F0 16 A5 07 85 EE A5 06 18 69 80 85 06 85 ED	584E
95B0:	90 A6 E6 07 E6 EE 4C 58 95 A5 08 18 69 28 85 ED	58F8
95C0:	85 06 85 08 A5 09 85 EE 85 07 18 90 8B 60 00 01	B059
95D0:	03 07 0F 1F 3F FE FC F8 F0 E0 C0 80 A2 08 06 EF	5418
95E0:	2A C9 07 90 04 E9 07 E6 EF CA D0 F2 60	933F

### **Si vous êtes néophyte**

Vérifiez que le programme Basic en mémoire a bien été sauvegardé sur disquette, puis tapez : NEW et HOME.

Il vous reste à passer en mode moniteur ce qui ne présente pas de difficulté. Tapez seulement un CALL - 151, suivi de RETURN.

Vous vous trouvez alors en présence de l'astérisque. Commencez à rentrer soigneusement vos codes. Ainsi, pour **FEN.HGR** (ci-dessus), tapez successivement :

\* 9500 : 20 7B... jusqu'à la fin de la ligne (ne tenez pas compte de la valeur 8E03, destinée

au contrôle, en utilisant la disquette SIGNATURE). Terminez par un RETURN et passez à la ligne suivante, à traiter de la même manière.

Lorsque vous aurez terminé la saisie, vous pourrez sauver immédiatement votre travail en tapant **BSAVE FEN.HGR,A\$9500,L\$ED...** et RETURN.

Pour revenir en mode normal, CONTROLE-C et RETURN. Ce retour peut précéder la sauvegarde sur disquette... ou venir après. C'est sans importance.

*J'ai oublié : avant de saisir un programme, quel qu'il soit, essayez toujours de lire le catalogue de votre disquette. Si cela se révèle impossible, c'est que le DOS n'est pas en mémoire... et tous vos efforts de saisie se révéleraient vains !*

NESTOR.

Pour commander la disquette

## **SIGNATURE**

utilisez le bulletin de la page 75. Offrez-vous aussi nos éditions spéciales (**ROUTINES LM POUR 6502, CLINS D'OEIL AU 6502**) : elles vous aideront à vous familiariser avec le langage machine... celui qui reste évidemment le plus rapide de tous les langages !



# Yvan KOENIG répond à nos lecteurs

## HORLOGE IIc ou IIe et RND

**Questions :** 1° Comment accéder à l'horloge du IIc ou du IIe ?  
2° Est-il possible d'améliorer le RND de l'Apple ?

Elie H. (51000 METZ)

- R** 1° Il n'y a pas d'horloge dans le IIe ou IIc, et vous ne risquiez donc pas d'en trouver une.
- 2° De nombreuses tentatives ont été faites pour améliorer le RND de l'APPLE. La meilleure réponse a été publiée dans *Call A.P.P.L.E.* en Janvier 1983, pages 29 à 34. L'adresse de *Call A.P.P.L.E.* est :  
290 SW 43 rd St, Renton, WA 98055, USA.

## LA COULEUR SUR IMAGEWRITER 2

**Question :** Existe-t-il des logiciels permettant l'impression couleur sur l'IMAGEWRITER 2 ? Damien S. (81000 CASTRES)

- R** Plusieurs logiciels permettent l'impression couleur sur IMW2. Il me semble que c'est le cas des versions récentes de PRINT SHOP. Ce qui est certain, c'est que EXTASIE, produit français, (cocorico !) permet l'impression couleur, tout comme BLAZING PADDLES et surtout DAZZLE DRAW. Un article de NIBBLE (décembre 1986) testait ces 2 derniers produits.

- EXTASIE est édité par APPLE France.
- BLAZING PADDLES par Baudville, 1001 Medical Park Dr., S.E. Grand Rapids, MI 49506 U.S.A. (tarif catalogue : \$49.95)
- DAZZLE DRAW par Broderbund Software, 17 Paul Dr. San Rafael, CA 94903 U.S.A. (tarif catalogue : \$59.95).

Il me semble avoir vu DAZZLE DRAW dans une boutique SIVEA.

## FAIRE.F1 ET PRODOS

**Question :** Intéressé par votre revue, j'ai acquis ses premiers numéros, et j'ai voulu utiliser la fonte que vous proposez page 22 du n°1. J'ai recopié les divers programmes d'accompagnement, sans comprendre. A noter que mon Apple refuse d'appliquer l'instruction  $D\$ = CHR\$(13) + CHR\$(4)$ , mais le plus ennuyeux est que l'instruction CALL 24751 (instruction 870 du programme FAIRE.ECRAN) n'est pas exécutée. On s'en rend compte par deux PEEKs aux adresses 54 et 55 qui auraient dû être chargées de \$BB à \$60 respectivement. A ce moment-là, la machine me rend la main, et il n'y a plus rien à faire. Vous est-il possible de m'aider ou dois-je faire mon deuil des efforts déjà accomplis ?

**Autre question :** j'ai vu l'assembleur ProCODE en vente à la FNAC, mais destiné au processeur 65C02. En existe-t-il une version pour le 6502 que je possède ?

R. S. (92370 CHAVILLE)

- R**
- FAIRE.1 (la police de caractères graphiques) et FAIRE.LM (groupe de routines qui permet entre autres d'utiliser la police pour écrire sur écran graphique) sont des programmes sous DOS 3.3 ce qui est la cause de vos problèmes. A ses débuts, Tremplin Micro ne publiait QUE des programmes DOS 3.3. Ce n'est pas la machine qui refuse  $D\$ = CHR\$(13) + CHR\$(4)$ , c'est ProDOS. Remplacez cette instruction par  $D\$ = CHR\$(4)$  et ça ira mieux. Mais il faudra aussi intervenir dans la routine HEX. En \$60AF essayez de remplacer la routine initiale par :  
\$60AD : A9 BB LDA £\$BB  
\$60AF : 8D 30 BE STA \$BE30  
\$60B2 : A9 60 LDA £\$60  
\$60B4 : 8D 31 BE STA \$BE31  
\$60B7 : 60 RTS
  - Comme vous ne dites pas quel est votre ordinateur, je ne peux pas vous faire une réponse précise sur ProCODE. Cet assembleur est capable de traiter des codes 65C02 mais le programme lui-même ne les utilise pas et peut donc fonctionner sur les IIe vieilles ROMS. Si votre machine est un II+, pas question d'utiliser ProCODE !

## ATTENTION AU GET !

**Question :** J'ai essayé, sans succès, de mélanger deux de vos programmes :

- La mémoire de votre Apple (TM n°1 page 42) et
- Copie d'écran 80 colonnes (TM n°9 page 19).

J'ai déplacé le programme de copie d'écran en \$6000 et je l'appelle par CALL 24576 (ligne 110 cf. listing). Utilisé seul par RUN 100, le programme fonctionne très bien en copie, mais utilisé après le programme d'examen de mémoire, cela ne va plus. Il ne me semble pas que les deux programmes utilisent des adresses communes.

Alors ?... Ai-je commis une erreur ou les deux programmes sont-ils incompatibles ? Dans ce cas, comment faire une copie d'écran pour "La mémoire de votre Apple" ?

Christian L. (44300 NANTES)

- R**
- A mon avis, il n'y a pas incompatibilité entre les deux programmes que vous essayez d'accoupler. Il y a simplement un oubli de votre part. Après un GET, il faut TOUJOURS faire un PRINT pour que le système d'exploitation puisse reconnaître les commandes qui lui sont destinées. Mettez un simple PRINT entre le GET H\$ et le IF de la ligne 40 et ça devrait aller mieux.

## SON (N°5) ET WAIT

Michel L. (68150 RIBEAUVILLE) a des problèmes avec la routine SON de Tremplin Micro n°5... et s'interroge à propos de WAIT...

**R** Notre ami R. THIBERT s'est planté dans la définition de HIMEM sous ProDOS. La valeur mise en place par la ligne 10 est doublement incorrecte.

1° Elle n'est pas en limite de page alors que ProDOS exige que HIMEM soit un multiple de 256 ce qui n'est pas le cas ici.

2° Il aurait de toute façon dû se placer 4 pages en-dessous de la routine, pour laisser de la place pour le buffer I/O de ProDOS.

Etablissez HIMEM:35840, et je pense que ça ira mieux. Ce n'est pas la meilleure solution pour protéger une routine sous ProDOS mais la seule bonne formule supposerait une réécriture complète que je pense entreprendre sous peu.

Votre question concernant WAIT est judicieuse, il n'y a AUCUNE différence entre WAIT-16384,128,127 et WAIT-16384,128.

Je recopie la définition que donnait le manuel de référence BASIC APPLESOFT de mon vieux II+.

WAIT 16000,255,0 permet d'insérer dans le programme une pause conditionnelle. Le premier argument est l'adresse décimale de l'emplacement en mémoire à tester pour voir si certains bits ont la valeur 1 et certains bits ont la valeur 0. Chaque bit de l'équivalent binaire du second argument décimal indique si vous êtes ou non intéressé au bit correspondant à l'emplacement en mémoire : la valeur 1 signifie que vous êtes intéressé, la valeur 0 signifie que vous devez ignorer ce bit. Chaque bit de l'équivalent binaire du troisième argument décimal indique quel état vous attendez (WAIT) pour le bit correspondant à l'emplacement en mémoire : la valeur 1 signifie que le bit doit avoir la valeur 0, la valeur 0 signifie que le bit doit avoir la valeur 1. S'il n'y a pas de troisième argument, on suppose la valeur 0. Si l'un quelconque des bits indiqués par un bit de valeur 1, dans le second argument correspond à l'état indiqué pour ce bit par le bit correspondant du troisième argument, l'instruction WAIT est terminée.

Dans votre cas, (128 = %10000000) indique que seul le bit 8 est utile et donc, 127 = %01111111 ou 0 = %00000000 est équivalent.

## RUBANS SCRIBE

Claude G. (13014 MARSEILLE) ne sait où se procurer des rubans pour son imprimante SCRIBE...

**R** Je me suis renseigné auprès d'APPLE FRANCE, chose que votre distributeur aurait dû faire. Normalement vous devez pouvoir vous procurer des rubans SCRIBE chez MICRO.VALLÉE 83/85 rue de Javel 75015 PARIS.

## HORLOGE

Très intéressé par "Horloge" (TM5 & TM12), je retrouve le même défaut sur les deux programmes, à savoir un retard de 9'06" par heure entre mon APPLE et l'horloge parlante. Néophyte en matière de langage machine je vous serais reconnaissant de bien vouloir m'éclairer à ce sujet.

Rodolphe De G. (27000 EVREUX)

**R** HORLOGE ne respecte pas les normes ProDOS. Cependant, en ce qui concerne le problème de la vitesse de l'horloge, le remède est simple. Tout se passe bien sur IIe mais sur IIc, le rafraîchissement écran se faisant à une vitesse différente, il faut corriger un octet au programme.

Tremplin n°7 avait publié dans le courrier des lecteurs le patch pour What time... il suffisait de faire \$386:32 (au lieu de 3B).

Pour HORLOGE je vous propose une formule plus élaborée qui permet au programme de régler lui-même sa vitesse selon la machine.

```
$9391 : 20 C2 95
```

```
$95C2 : A9 3B      LDA £$3B
        AE C0 FB   LDX $FBC0  est-ce un IIc ?
        D0 02     BNE $95CB   non
        A9 32     LDA £$32
```

```
$95CB : 8D 2D 94  STA $942D  règle la vitesse
        4C C5 93  JMP $93C5   teste présence carte souris
```

**BSAVE HORLOGE,A\$9000,E\$95D0 (ou L\$5D1)**

Avec le programme modifié ou non, ne tenez pas compte des indications de l'auteur et faites HIMEM:35840. Cela évitera l'écrasement de la routine de saisie date/heure par les buffers de ProDOS.

## AVIS AUX UTILISTATEURS D'IMPRIMANTES EPSON

La carte interface parallèle EPSON réf. 8132 (celle qui porte les roms APL.B ou D ou E ou YK) est parfaitement incompatible avec APPLEWORKS.

Technology Resources a pris une initiative louable à ce sujet en proposant le remplacement GRATUIT de ces cartes par un nouveau modèle compatible APPLEWORKS et portant la référence 8133.

Pour obtenir l'échange il faut renvoyer votre carte à :

TECHNOLOGY RESOURCES  
service pièces détachées  
80/86 av. Louis Roche  
EVOLIC C-201 — 92230 GENEVILLIERS  
Tél. : (16-1) 47.92.01.13

# API NEWS

DES NOUVEAUTES POUR VOTRE APPLE II, IIe, IIc \*



EDUCATIFS - JEUX - PROGRAMMATION

\* APPLE est une marque déposée d'APPLE COMPUTER, INC

## COLLECTION OSCAR

Logiciels éducatifs pour les 3 à 8 ans, faisant appel pour la plupart à la SYNTHÈSE VOCALE sur matériel standard. Graphismes splendides.

### 1 . L'ALPHABET DE L'OURS OSCAR

Apprentissage de l'alphabet par un jeune enfant à travers 5 applications différentes. SYNTHÈSE VOCALE : lettres énoncées à voix haute, enfant guidé par des messages sonores. Nécessite deux lecteurs et un joystick. ... 185 F

### 2 . L'ALBUM DE COLORIAGE

De nombreux dessins à colorier. Possibilité pour l'enfant de sauvegarder ses dessins et de les rappeler. Création d'albums sur papier par transfert sur imprimante (imagewriter). Nécessité d'un joystick - Livré avec un mini-album et dix feutres. ... 185 F

### 3 . L'ALBUM ENCHANTE

Logiciel d'éveil pour les tous petits. Reconnaissance d'animaux, d'objets, de formes, etc ... à travers un album qui parle ! Comporte de plus un puzzle. ... 185 F

### 4 . OSCAR PUZZLES

Un logiciel captivant pour l'enfant et très simple d'emploi. Deux niveaux de difficulté - Planches superbes - Joystick ou clavier. ... 185 F

### 5 . OSCAR LECTURE 1

Apprentissage de la lecture par un jeune enfant. Reconnaissance d'images, de mots écrits à l'écran ou énoncés à voix haute. Large appel à la synthèse vocale. ... 185 F

### 6 . CONTES DE L'AFRIQUE NOIRE

Lecture d'un conte à voix haute. Idéal pour les tous petits ne sachant pas encore lire. Pour les plus grands, questions sur le conte, vocabulaire, etc ... Nécessité de deux lecteurs. ... 65 F

## JEUX

### 7 . LE PRIVE

Une enquête policière délicate : qui a tué M. CHABOT le restaurateur ? Les mobiles ne manquaient pas ! - Graphismes superbes - Une nouveauté : le bulletin liaison de réponse. ... 185 F

## PROGRAMMATION

### 8 . BASIC EXPRESS

Apprenez rapidement le langage de votre Apple. Une méthode simple, accessible à tous, enfants et adultes. Manuel, fiches de progression, disquettes. ... 279 F

BON DE COMMANDE A RETOURNER A :

API  
12 Chemin du Petit Etang  
86000 POITIERS

Règlement par chèque bancaire ou C.C.P -  
Pas d'envoi contre remboursement.

Nom : .....

ADRESSE : .....

DATE : .. .. .

SIGNATURE :

COCHEZ : 1 2 3 4 5 6 7 8



# Votre bibliothèque INFORMATIQUE

par NESTOR

- **ALGORITHMES ET MATHÉMATIQUES**  
(J.-M. Chauveau et S. Weber)

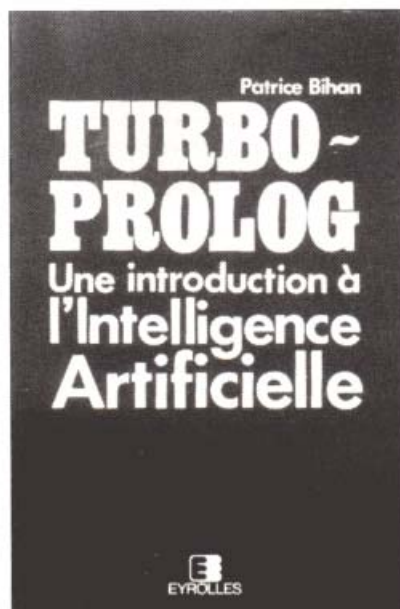
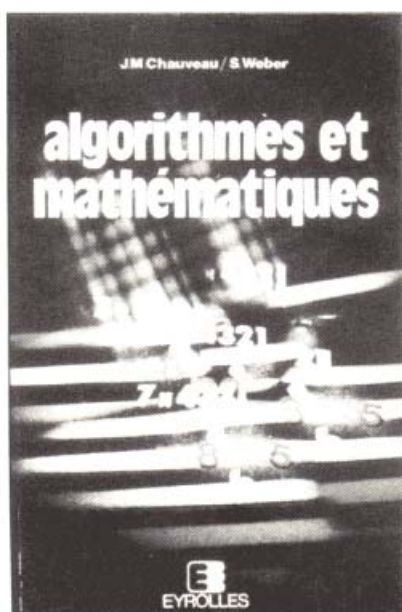
Maîtriser l'informatique... c'est assurément le souhait de tout lecteur de *Tremplin Micro*. Mais c'est d'abord être en mesure :

- de savoir ce que l'on veut faire ;
- de savoir comment transmettre son désir à l'ordinateur.

Les auteurs de ce livre préconisent d'accéder à l'informatique par le biais des mathématiques. Pourquoi pas ?

D'abord dérouté par une approche un peu abrupte des conventions d'écriture des procédures, le lecteur non matheux se retrouvera dans son élément avec des exemples simples, en Basic classique. De bons algorithmes de tri et des pages passionnantes sur la conception et la réalisation d'algorithmes en probabilité et en statistiques. Une information mathématique type DEUG est conseillée pour tirer le meilleur profit de cet ouvrage, mais les exemples en Basic peuvent par contre être compris par des programmeurs moyens.

EYROLLES, 61, bd Saint-Germain, 75005 PARIS



- **TURBO-PROLOG**  
*Une introduction à l'Intelligence Artificielle*  
(P. Bihan)

TURBO-PROLOG, c'est bien sûr un produit BORLAND INTERNATIONAL. Plus de 50 000 concepteurs, programmeurs, ingénieurs et autres professionnels se servent de ce langage pour concocter des applications faisant appel à l'Intelligence Artificielle. Il convient de noter que BORLAND offre maintenant aux utilisateurs de TURBO-PROLOG une puissante boîte à outils : TURBO-PROLOG TOOLBOX. Plus de 80 outils et 40 exemples avec leur code source : ah ! les veinards ! A quand l'équivalent sur l'Apple IIGS ?

Mais revenons au bouquin de Patrice Bihan. Il est à lire devant un clavier de PC. En fait, tous les exemples gagneront à être écrits, puis exécutés. L'introduction n'est sûrement pas indispensable aux utilisateurs pratiquant parfaitement l'anglais (je ne crois pas avoir vu une version française du manuel BORLAND), mais elle sera utile à tous les autres. Quant aux 60 petites

applications... à nos claviers et place à l'imagination !

EYROLLES, 61, bd Saint-Germain, 75005 PARIS

- **C PAR LA PRATIQUE**  
(Craig Bolon)

Cet ouvrage a été traduit de l'américain par Lionel Eppelé. Faut-il féliciter l'auteur — Craig Bolon — ou le traducteur de la clarté du style ? Toujours est-il que *C PAR LA PRATIQUE* est un manuel agréable à lire et facile à comprendre. Si vous n'entendez rien à la programmation, essayez tout de même de survoler les principales instructions du Basic, mais sachez que ce n'est pas indispensable. Vous assimilerez ici, en plus de 500 pages et en vingt chapitres, non seulement les bases de la programmation en C, mais comment concevoir des applications relativement importantes. Ce bouquin n'est pas destiné aux utilisateurs de l'Apple IIGS, mais gageons que ceux-ci y puiseront, s'ils désirent s'initier au langage de l'avenir qu'est le C, de très utiles informations : bases du langage, types de données, utilisation des structures de tableaux, problèmes de récursivité, opérateurs, calculs en virgule flottante, bibliothèques au format Unix, etc. C impressionnant, non ?

SYBEX, 6-8, impasse du Curé — 75018 PARIS  
(552 pages — 248 F TTC)



# Chasseur d'Images

**Chaque mois,  
le meilleur  
de la  
technique  
et de la  
pratique  
photo !**



**Chez votre  
marchand de journaux !**