Ouvrage collectif publié sous la direction
de Guy-HACHETTE, fondateur de Tremplin Micro

Programmes LM

65CØ2 et 65C816

Argos, Géo, Nestor, Clément Renard

UNE ÉDITION DE TREMPLIN MICRO

Ouvrage collectif publié sous la direction
de Guy-HACHETTE, fondateur de Tremplin Micro

Programmes LM

65CØ2 et 65C816

Argos, Géo, Nestor, Clément Renard

UNE ÉDITION DE TREMPLIN MICRO

EDITIONS JIBENA — TREMPLIN MICRO —
La Petite Motte — Senillé — 86100 CHÂTELLERAULT

ISBN 2-901124-26-7

OUI
au langage machine !

Vous avez choisi de vous initier à la pratique du langage machine... et vous avez raison : c'est celui que votre Apple comprend le mieux... même si ce n'est apparemment pas le plus simple à utiliser.

Les recueils que nous publions dans cette collection ne prétendent pas être des modèles du genre. Ils reprennent la plupart des petites routines parues dans ***TREMPLIN MICRO****, mais en les complétant parfois par d'autres. Ils comportent aussi un certain nombre d'inédits.*

Nous tenons pour acquis que votre machine est au moins équipée d'un 65C02 autorisant les nouvelles instructions. Si vous disposez d'un Apple IIGS, l'émulation sera parfaite.

Quant à ProDOS, c'est devenu LE système. Nous en restons ici à ProDOS 8, mais il est certain que ProDOS 16 prendra bientôt le relai.

Mieux vous connaîtrez le premier, mieux cela vaudra pour utiliser pleinement le second.

Bonne programmation ! ***Guy-Hachette.***

DANS LA MÊME COLLECTION

CLINS D'ŒIL AU 65C02 DE L'APPLE

Plus de 40 programmes parus ou non dans *Tremplin Micro*, et facilement utilisables à partir du Basic. La plupart d'entre eux sont compatibles avec le 65C02. A quelques rares exceptions près, ces routines fonctionnent aussi sur l'Apple IIGS.

ROUTINES LM POUR 65C02 ET 6502

Plus de 45 programmes (la plupart ont paru dans *Tremplin Micro*, mais d'autres sont inédits) permettent de bien comprendre comment passer du Basic au langage machine.

Des points forts comme la *Création d'une police de caractères graphiques*, mais aussi plusieurs dizaines de routines dont la bonne compréhension est facilitée par de nombreux commentaires.

NOUVELLES ROUTINES POUR LE 65C02 (et le 65C816)

Plus de 40 programmes (extraits ou non de *Tremplin Micro*). On a particulièrement apprécié *FaireEcran80*, *Multi.Rech*, etc.

Ces trois ouvrages sont fournis avec une disquette d'accompagnement.

TREMPLIN MICRO — LA PETITE MOTTE — SENILLÉ — 86100 CHATELLERAULT

INDEX GÉNÉRAL

CATALOGUE DE VOTRE DISQUETTE

ESCAPE POUR SORTIR

PAGE NUMÉRO 1

A. CREDIR ..... /CREDIR
B. MULTIPLIER ..... /MULTIPLIER
C. CLAVIERØ ..... /CLAVIERØ
D. CHUTE.POMMES ..... /CHUTE.POMMES
E. FICDIR ..... /FIC/FICDIR
F. FICDIR1 ..... /FIC/FICDIR1
G. ALLO.EXEC ..... /ALL/ALLO.EXEC
H. ALIGN.DR4Ø ..... /ALI/ALIGN.DR4Ø
I. ALIGN.DR8Ø ..... /ALI/ALIGN.DR8Ø
J. ALIGN MIXTE ..... /ALI/ALIGN.MIXTE
K. ALIGN.BASØ ..... /ALI/ALIGN.BASØ
L. ALIGN.BAS1 ..... /ALI/ALIGN.BAS1
M. TRANSNBR ..... /TRANSNBR
N. JEU.ALTERNE ..... /JEU.ALTERNE
O. COPHGR.B ..... /COPIE.HGR/COPHGR.B
P. OCCUPATION ..... /OCCUP/OCCUPATION
Q. OCCUPAMP ..... /OCCUP/OCCUPAMP
R. OCCUP.RUN ..... /OCCUP/OCCUP.RUN
S. SERIE.ALEA ..... /ALEA/SERIE.ALEA
T. SERIE.ALEA2 ..... /ALEA/SERIE.ALEA2
U. TABBUF.DEMO ..... /TABBUF.DEMO

LETTRE-CLE CHOISIE : SUITE :

PAGE NUMÉRO 2

A. TABBUF.GS ..... /TABBUF.GS
B. COPHGR.B1 ..... /COPIE.HGR/COPHGR.B1
C. QUOI.EST.OU ..... /QUOI.EST.OU
D. QUOI4Ø.8Ø ..... /QUOI4Ø.8Ø
E. ECHANGE ..... /ECHANGE
F. HGR.UPBAS ..... /HGR/HGR.UPBAS
G. HGR.DOWNBAS .... /HGR/HGR.DOWNBAS
H. HGR.RBAS ..... /HGR/HGR.RBAS
I. HGR.RNT1B ..... /HGR/HGR.RTN1B
J. PRT.USING ..... /USING/PRT.USING
K. PRT.USVIRG ..... /USING/PRT.USVIRG
L. PRT.USVAR ..... /USING/PRT.USVAR
M. PRT.USPIL ..... /USING/PRT.USPIL
N. PRT.USPIL1 ..... /USING/PRT.USPIL1
O. VIRGULE ..... /VIRGULE
P. MEL.ESSAI ..... /MEL/MEL.ESSAI
Q. MEL.BAS ..... /MEL/MEL.BAS
R. MIN.BAS ..... /MIN/MIN.BAS
S. PRO.DEMO ..... /PRO/PRO.DEMO
T. DIVISION ..... /DIVISION
U. QUATRE.OPER ..... /QUATRE.OPER

LETTRE-CLE CHOISIE : SUITE :

PAGE NUMÉRO 3

A. ALLER.RETOUR ..... /ALLER.RETOUR
B. GETNUM ..... /GETNUM
C. MIDNUM ..... /MIDNUM
D. ADR.DEMO ..... /ADRCHG/ADR.DEMO
E. RND ..... /RND
F. LØ ..... /ECRLETTRES/LØ
G. L1 ..... /ECRLETTRES/L1
H. BANDE.PASSANTE ..... /BANDE.PASSANTE
I. PROBLEME ..... /PROBL/PROBLEME
J. SLMØ ..... /PROBL/SLMØ
K. SLM1 ..... /PROBL/SLM1
L. SLM2.BAS ..... /PROBL/SLM2.BAS
M. DAT.AFF ..... /DAT/DAT.AFF
N. DAT.VAR ..... /SPECIAL/DAT.VAR
O. TRI.B ..... /TRI/TRI.B
P. TAB ..... /TAB
Q. POPFT ..... /POPFT
R. UN × DEUX ..... /UN×DEUX
S. MLI ..... /MLI
T. BLOLIB.A ..... /SECT/BLOLIB.A
U. BLOLIB.V ..... /SECT/BLOLIB.V

LETTRE-CLE CHOISIE : SUITE :

PAGE NUMÉRO 4

A. ASCII.BCD ..... /ASCII.BCD
B. VERT ..... /VERT

LETTRE-CLE CHOISIE : SUITE :

ATTENTION ! la disquette ne comporte ni ProDOS ni BASIC.SYSTEM, mais vous avez tout juste la place pour installer ces deux fichiers. Si vous effacez votre disquette par mégarde, vous retrouverez les mêmes programmes sur la face protégée.

CREDIR

A quoi bon ce programme, alors qu'il est tellement facile, sous ProDOS, en mode direct ou à partir du Basic, de créer un sous-catalogue par CREATE/PREFIX/NOM ? Simplement pour montrer l'un des moyens utilisés pour obtenir le même résultat à partir d'une routine LM. Peut être utilisé par **BRUN CREDIR**.

```
300 :  20 58 FC     JSR FC58      Le Home habituel vide l'écran.
303 :  A0 00        LDY £00
305 :  B9 53 03     LDA 0353,Y    Affichage du texte de l'INPUT (CREER DIR : ).
308 :  F0 06        BEQ 0310      Le message se termine par un 0, d'où le BEQ
30A :  20 ED FD     JSR FDED      $310 qui permet de sortir de la boucle.
30D :  C8           INY
30E :  D0 F5        BNE 0305
310 :  20 6F FD     JSR FD6F      INPUT : à la sortie, la longueur est dans X. Si
313 :  8A           TXA           elle est nulle, terminé !
314 :  F0 14        BEQ 032A
316 :  8D 5F 03     STA 035F      On la stocke ici pour le MLI.
319 :  BD FF 01     LDA 01FF,X    Nom du sous-catalogue (/PREFIX/NOM)
31C :  9D 5F 03     STA 035F,X    transféré où le MLI va le lire.
31F :  CA           DEX
320 :  D0 F7        BNE 0319
322 :  20 00 BF     JSR BF00      C'est là que ça se passe. Point d'entrée.
325 :  C0           C0            $C0 pour CREATE.
326 :  3E 03        033E          Adr. du point d'entrée de la liste des paramètres.
328 :  D0 01        BNE 032B      On ne fera le saut qu'en cas d'erreur.
32A :  60           RTS           Terminé et correct.
32B :  20 DA FD     JSR FDDA      Le numéro de l'erreur est affiché.
32E :  A0 00        LDY £00
330 :  B9 4A 03     LDA 034A,Y    Puis le message : ERREUR.
333 :  F0 06        BEQ 033B
335 :  20 ED FD     JSR FDED
338 :  C8           INY
339 :  D0 F5        BNE 0330
33B :  4C 3A FF     JMP FF3A      Un petit bip pour finir.
33E :  07 5F
340 :  03 C3
342 :  0F 00 00 0D  On indique successivement : le nombre de paramètres (7),
346 :  00 00        l'adresse du nom (2 octets), le code d'accès ($C3, 1 octet), le type
348 :  00 00        (0F pour DIR, 1 octet), 2 octets vides, le type de sauvegarde (0D, 1
                    octet) et 4 octets pour la date et l'heure (inutilisés).
```

BSAVE CREDIR,A$300,L$5F

TEXTES

```
34A :  BA A0 C5 D2 D2 C5 -      : ERRE
350 :  D5 D2 00 C3 D2 C5 C5 D2 - UR.CREER
358 :  A0 C4 C9 D2 BA A0 00 -    DIR: .
```

■

MULTIPLIER

Petite multiplication par addition (jusqu'à 255)

```
100 FOR I = 768 TO 793: READ R: POKE I,R: NEXT
105 DATA 32,245,230,202,134,6,32,245,230,138,160,0,24,101,6,144,1,200,202,208,247,
    170,152,76,36,237
110 TEXT : HOME : PRINT CHR$ (21)
115 INPUT "A   B >  ";A$
120 FOR I = 1 TO LEN (A$): IF MID$ (A$,I,1) < > " " THEN NEXT
125 A = VAL ( LEFT$ (A$,I)): IF I > = LEN (A$) THEN 110
130 B = VAL ( RIGHT$ (A$, LEN (A$) - I))
135 IF A = 0 OR B = 0 OR (A > 255 OR B > 255) THEN 110
140 PRINT : PRINT : CALL 768,A,B: PRINT
145 VTAB 22: PRINT "ENFONCEZ UNE TOUCHE  ";: GET R$: PRINT : IF R$  < > CHR$ (27)
    THEN 110
150 VTAB 22: PRINT "(M)ENU DE DISQUETTE —›  ";: GET R$: PRINT
155 HOME : IF R$ = "M" THEN PRINT CHR$ (4)"RUN MENU"
```

```
300 :   20  F5  E6      JSR  $E6F5
303 :   CA              DEX
304 :   86  06          STX  $06
306 :   20  F5  E6      JSR  $E6F5
309 :   8A              TXA
30A :   A0  00          LDY  £$00
30C :   18              CLC
30D :   65  06          ADC  $06
30F :   90  01          BCC  $0312
311 :   C8              INY
312 :   CA              DEX
313 :   D0  F7          BNE  $030C
315 :   AA              TAX
316 :   98              TYA
317 :   4C  24  ED      JMP  $ED24
```

A — GETBYTC saute un caractère et évalue l'expression en passant par GETBYT ($E6F8) et CONINT ($E6FB). Le résultat est dans X et dans $A1.

B — Même topo. On passe X dans A.

Y = 0 pour initialiser la boucle d'addition. Annulation de la retenue. On additionne B fois le nombre (A - 1) à l'Accumulateur (lequel contient B au départ). Exemple : 4 × 3 = (3 dans $6) 3 + (3 + 3 + 3 + 3)

A dans X

Y dans A pour que...

... LINPTR affiche le résultat.

Deuxième formule

```
160 END
165 DATA 32,245,230,134,6,32,245,230,169,0,168,24,101,6,144,1,200,202,208,247,170,
    152,76,36,237,0: REM Seconde formule
```

```
300 :   20  F5  E6      JSR  $E6F5
303 :   86  06          STX  $06
```

A est dans $6.

```
305 :  20  F5  E6      JSR  $E6F5
308 :  A9  00          LDA  £$00
30A :  A8              TAY
30B :  18              CLC
30C :  65  06          ADC  $06
30E :  90  01          BCC  $0311
310 :  C8              INY
311 :  CA              DEX
312 :  D0  F7          BNE  $030B
314 :  AA              TAX
315 :  98              TYA
316 :  4C  24  ED      JMP  $ED24
```

B est dans X (nombre de fois à multiplier). Parties basse et haute du futur total à 0.

Annulation de la retenue.

Si retenue, partie haute (Y) incrémentée.

X = X - 1 (boucle).

Affichage. ■

FICHIER À ACCÈS DIRECT

De nombreux débutants s'interrogent sur les règles qui déterminent la longueur d'un enregistrement dans un fichier à accès direct. Comment la calculer ? C'est simple : elle est égale à la longueur de la donnée la plus longue + 1 (le RETURN qui marque la fin de chaque enregistrement). Mais il existe une exception : quand on désire mémoriser une donnée comportant une ponctuation interdite par l'Applesoft, il faut ouvrir les **guillemets** — CHR$ (34) — en tête de l'enregistrement... et **majorer la fameuse longueur d'une unité**.

EXPÉRIENCE : Testez la démo ci-dessous (sous ProDOS exclusivement). Elle crée un fichier à accès direct intitulé ESSAI, y écrit, dans l'enregistrement n°1, le contenu de la variable A$ (**Vrai, oui**)... précédé des **guillemets.** Ensuite, grâce aux possibilités de ProDOS, elle le recopie mais en **mode binaire**, à partir de l'adresse **$300**, l'efface sur la disquette (bon débarras !), et vous permet enfin de l'examiner sans passer par le fatigant **CALL -151** !

```
10 TEXT : HOME :D$ = CHR$ (4): PRINT
   D$"PR£3": PRINT : LIST
15 G$ = CHR$ (34): REM Guillemets
20 L = 11: REM Longueur de l'enregistrement
25 N = 1: REM Numéro d'enregistrement
30 T$ = "ESSAI": REM Titre du fichier
35 A$ = "Vrai, oui"
40 PRINT D$"OPEN";T$;",L";L
45 PRINT D$"WRITE";T$;",R";N
50 PRINT G$ + A$
55 PRINT D$"CLOSE";T$
60 PRINT    D$"BLOAD";T$;",TTXT,A$300":
   PRINT D$"DELETE";T$
65 E$ = "300.317 D823G"
70 FOR I = 1 TO LEN (E$): POKE 511 + I, ASC
   ( MID$ (E$,I,1)) + 128: NEXT : POKE 72,0:
   CALL - 144: PRINT
75 VTAB 24: HTAB 32: PRINT "(M)ENU DE
   DISQUETTE ";: GET R$: VTAB 20: PRINT :
   IF R$ = "M" THEN PRINT D$"RUNMENU"
```

Fichier mémorisé sous le titre **FICDIR**

```
300 :  00 00 00 00 00 00 00 00
308 :  00 00 00 22 56 72 61 69
310 :  2C 20 6F 75 69 0D 00 00
```

11 octets réservés pour l'enregistrement 0 (inutilisé), puis 11 octets occupés par **"Vrai, oui"** + RETURN (0D). Le compte est bon ! *(voir aussi page 11)*

CLAVIERO

Petite routine sans prétention, destinée à tester la frappe d'une touche... additionnée à celle de POMME OUVERTE ou POMME FERMÉE. On peut indifféremment frapper le caractère + POMME OUVERTE (ou FERMÉE) ou bien PO (ou PF) + le caractère. Par contre, il ne faut pas enfoncer les deux touches à la fois (du moins avec ce programme).

```
10  TEXT : NORMAL : HOME : PRINT CHR$ (21):D$ = CHR$ (4)
20  FOR I = 768 TO 840: READ R: POKE I,R: NEXT
30  DATA  32,58,255,173,97,192,48,5,172,98,192,16,246,44,0,192,16,251,174,0,192,44,
    16,192,218,90,169,23,32,91,251,169,0,133,36,32,156,252
35  DATA  169,208,32,237,253,169,207,122,192,128,144,2,169,198,32,237,253,169,171,
    32,237,253,250,218,169,0,32,36,237,250,224,155,208,184,96
40  VTAB 1: INVERSE : PRINT "POMME OUVERTE OU FERMÉE, PLUS UNE TOUCHE":
    NORMAL
45  VTAB 7: PRINT "PO ou PF + ESCAPE pour terminer"
50  CALL 768: VTAB 20
55  PRINT : VTAB 20: PRINT "(M)ENU DE DISQUETTE  (E)NCORE  (T)ERMINE ";: GET R$:
    IF R$ > "T" THEN R$ = CHR$ ( ASC (R$) - 32)
60  PRINT R$: IF R$ = "E" THEN RUN
65  IF R$ = "T" THEN HOME : END
70  IF R$ = "M" THEN PRINT D$"RUNMENU"
75  GOTO 55
```

```
300 :  20 3A FF    JSR $FF3A     BIP.
303 :  AD 61 C0    LDA $C061     Lecture PO (Accumulateur).
306 :  30 05       BMI $030D     Pressée ? saut.
308 :  AC 62 C0    LDY $C062     Lecture PF (Registre Y).
30B :  10 F6       BPL $0303     Pas pressée ? retour en arrière.
30D :  2C 00 C0    BIT $C000   ] Si aucune touche n'a été pressée, retour
310 :  10 FB       BPL $030D   ] case-départ.
312 :  AE 00 C0    LDX $C000     Clavier dans le registre X.
315 :  2C 10 C0    BIT $C010     Clavier remis à zéro.
318 :  DA          PHX         ]
319 :  5A          PHY         ] X et Y sur la pile.
31A :  A9 17       LDA £$17    ]
31C :  20 5B FB    JSR $FB5B   ] Sur la ligne 24.
```

```
031F :  A9 00       LDA £$00      ] HTAB 1.
0321 :  85 24       STA $24       ]
0323 :  20 9C FC    JSR $FC9C     CLREOL efface le bout de la ligne.
0326 :  A9 D0       LDA £$D0      ] P affiché.
0328 :  20 ED FD    JSR $FDED     ]
032B :  A9 CF       LDA £$CF      O dans l'Accumulateur.
032D :  7A          PLY           ] Y récupéré. S'il est plus petit que $80, c'est
032E :  C0 80       CPY £$80      ] bien pomme ouverte.
0330 :  90 02       BCC $0334     ]
0332 :  A9 C6       LDA £$C6      ] Sinon F dans l'Accumulateur...
0334 :  20 ED FD    JSR $FDED     ] et affichage.
0337 :  A9 AB       LDA £$AB      ] + affiché.
0339 :  20 ED FD    JSR $FDED     ]
033C :  FA          PLX           ] X récupéré et remis sur la pile presto.
033D :  DA          PHX           ]
033E :  A9 00       LDA £$00      ] LINPTR affiche la valeur décimale de X,A.
0340 :  20 24 ED    JSR $ED24     ]
0343 :  FA          PLX           ] X récupéré.
0344 :  E0 9B       CPX £$9B      ] Etait-ce ESCAPE ?
0346 :  D0 B8       BNE $0300     Non : on continue.
0348 :  60          RTS           Oui : on arrête. ■
```

FICHIER DIRECT SUITE

(de la page 9)

Un seul enregistrement peut contenir plusieurs données. Reprenons la démo de la page 9, mais en modifiant le contenu de quelques lignes :

```
20  L = 36: REM Longueur de l'enregistrement
35  A$ = "LAMBINE":B$ = "ADELE":C$ = "31 DECEMBRE 1899"
50  PRINT A$: PRINT B$: PRINT C$
65  E$ = "300.3F D823G"
```

Lançons le programme et examinons le contenu des enregistrements 0 et 1.

```
0300 :  00 00 00 00 00 00 00 00
0308 :  00 00 00 00 00 00 00 00
0310 :  00 00 00 00 00 00 00 00
0318 :  00 00 00 00 00 00 00 00
0320 :  00 00 00 00 4C 41 4D 42
0328 :  49 4E 45 0D 41 44 45 4C
0330 :  45 0D 33 31 20 44 45 43
0338 :  45 4D 42 52 45 20 31 38
0340 :  39 39 0D 00 00 00 00 00
0348 :  00 00 00 00 00 00 00 00
```

La partie en gras compte bien 36 octets réservés pour l'enregistrement n°0.

Dans l'enregistrement n°1, on constate que, chaque donnée est séparée de l'autre par un RETURN (0D), détail dont il faudra tenir compte... sans oublier les guillemets éventuels !

MÉMORISÉ SOUS LE TITRE **FICDIR1** ■

CHUTE-POMMES

Les pommes tombent. La récolte sera bonne : 960 très exactement... puisqu'elles remplaceront finalement tous les caractères de l'écran. Plusieurs remarques :

- Cette routine ne fonctionne (du moins avec les caractères souris) que si la carte 80 colonnes a été activée par **PR£3**, puis le mode 40 colonnes mis en service par un **PRINT CHR$ (17)**.
- Il est évidemment possible de remplacer la pomme pleine ($40) par une pomme vide ($41)... et par n'importe quel autre caractère souris.
- On obtient un très bel effet avec la flèche **$4A**.
- Si l'écran affiche déjà le caractère souris retenu, la routine cessera de traiter la colonne lorsque ce caractère atteindra la dernière ligne.

```
100 TEXT : NORMAL :D$ = CHR$ (4): PRINT D$"PR£3": PRINT CHR$ (17)
105 FOR I = 768 TO 824: READ R: POKE I,R: NEXT
110 DATA  32,174,239,165,146,41,63,201,40,176,245,168,169,64,133,6,162,0,138,32,71,248,
    177,38,72,165,6,145,38,104,133,6,232,224,24,208,237,160,39,162,40,177,38,201,64,208,
    1,202,136,16,246,202,16,202,76,58,255
115 LIST : GOSUB 160: CALL 768
120 GOSUB 160
125 PRINT D$"BLOAD CHTPM1"
130 PRINT : VTAB 24: HTAB 3: INVERSE : PRINT " SECONDE VERSION UN PEU PLUS
     RAPIDE ";: NORMAL : GOSUB 160: LIST : GOSUB 160: CALL 768: GOSUB 160
135 VTAB 22: CALL - 958
140 VTAB 24: PRINT "    (M)ENU DE DISQUETTE (E)NCORE ";: GET R$
145 IF R$ = "M" OR R$ = "m" THEN PRINT D$"RUNMENU"
150 IF R$ = "E" OR R$ = "e" THEN RUN
155 HOME : END
160 CALL - 198: POKE 49168,0: WAIT 49152,128: POKE 49168,0: RETURN
```

VERSION 0 (DATA)

```
300 :  20 AE EF    JSR $EFAE      RND.
303 :  A5 92       LDA $92        On trouve ici un nombre aléatoire.
305 :  29 3F       AND £$3F       Bits 7 et 6 à 0 (ça limite !).
307 :  C9 28       CMP £$28   ]   Plus grand ou égal $28 (40), c'est mauvais.
309 :  B0 F5       BCS $0300  ]   On va de 0 à 39.
30B :  A8          TAY            A dans Y.
30C :  A9 40       LDA £$40   ]
30E :  85 06       STA $06    ]   Caractère souris (pomme) dans $06.
310 :  A2 00       LDX £$00   ]
312 :  8A          TXA        ]   Initialisation du compteur de lignes (0 à 23).
```

```
313 :  20 47 F8     JSR  $F847     GBASCAL nous fournit l'adresse.
316 :  B1 26        LDA  ($26),Y   Lecture de la case-écran (colonne Y).
318 :  48           PHA            Sur la pile.
319 :  A5 06        LDA  $06       ┐ Pomme dans la case-écran.
31B :  91 26        STA  ($26),Y   ┘
31D :  68           PLA            ┐ Caractère-écran à la place de la pomme,
31E :  85 06        STA  $06       ┘ dans le tiroir $6.
320 :  E8           INX            ┐ Une ligne de plus. Jusqu'à $17 (23) inclus, on
321 :  E0 18        CPX  £$18      │ continue imperturbablement.
323 :  D0 ED        BNE  $0312     ┘
```

```
325 :  A0 27        LDY  £$27      ┐ On va successivement lire toutes les cases-
327 :  A2 28        LDX  £$28      │ écran de la dernière ligne et décrémenter X
329 :  B1 26        LDA  ($26),Y   │ chaque fois qu'une pomme sera rencontrée
32B :  C9 40        CMP  £$40      │ (colonne achevée).
32D :  D0 01        BNE  $0330     │ A l'issue de ce test, si X - 1 différent de $FF,
32F :  CA           DEX            │ un BPL permet de poursuivre l'exécution du
330 :  88           DEY            │ programme. Dans le cas contraire, on ter-
331 :  10 F6        BPL  $0329     │ mine sur un BIP.
333 :  CA           DEX            │
334 :  10 CA        BPL  $0300     │
336 :  4C 3A FF     JMP  $FF3A     ┘
```

CHTPM1

```
300 :  A9 28        LDA  £$28      ┐ Nombre de caractères par ligne dans $7 (40)
302 :  85 07        STA  $07       ┘ qui va nous servir de pèse-pommes.
304 :  20 AE EF     JSR  $EFAE     ┐ Nombre aléatoire obtenu dans les mêmes
307 :  A5 92        LDA  $92       │ conditions que précédemment (ce n'est pas
309 :  29 3F        AND  £$3F      │ génial, mais on ne cherche pas la grande
30B :  C9 28        CMP  £$28      │ vitesse !).
30D :  B0 F5        BCS  $0304     ┘
30F :  A8           TAY            Numéro de colonne dans Y (de 0 à 39).
310 :  A9 40        LDA  £$40      ┐ POMME SOURIS dans A. Y a-t-il une POMME
312 :  D9 D0 07     CMP  $07D0,Y   │ dans la Y colonne de la dernière ligne ?
315 :  F0 ED        BEQ  $0304     ┘ Si oui, vers un autre nombre aléatoire.
317 :  85 06        STA  $06       Pomme dans tiroir $6.
319 :  A2 00        LDX  £$00      Initialisation nombre de lignes (0 à 23).
31B :  A5 06        LDA  $06       ┐ Pomme sur la pile.
31D :  48           PHA            ┘
31E :  8A           TXA            ┐ Numéro de ligne dans A pour GBASCALC.
31F :  20 47 F8     JSR  $F847     ┘
322 :  B1 26        LDA  ($26),Y   ┐ Caractère-écran lu et mis dans le tiroir $6.
324 :  85 06        STA  $06       ┘
326 :  68           PLA            ┐ Pomme retournée sur la pile et poquée sur
327 :  91 26        STA  ($26),Y   ┘ l'écran.
329 :  E8           INX            ┐ X = X + 1. Si on est plus petit que $18 (24),
32A :  E0 18        CPX  £$18      │ on continue.
32C :  D0 ED        BNE  $031B     ┘
32E :  C9 40        CMP  £$40      ┐ A-t-on une pomme juteuse en mémoire ?
330 :  D0 D2        BNE  $0304     ┘ Non : on retourne illico au début.
332 :  C6 07        DEC  $07       ┐ Oui : on décrémente le pèse-pommes $7. Si
334 :  D0 CE        BNE  $0304     ┘ on n'est pas encore à 0, le travail continue.
336 :  4C 3A FF     JMP  $FF3A     Ou alors, c'est le BIP final !
```

BSAVE CHTPM1,A$300,L$39

ALLO ! RÉPERTOIRE TÉLÉPHONIQUE SUR APPLE

Sans doute utiliserez-vous cette routine pour créer un répertoire téléphonique personnel (contenance : 100 lignes, mais vous pouvez les augmenter à volonté). Peut-être envisagerez-vous de transformer son titre en **CAVE**..., puis de lui confier la liste de vos bonnes bouteilles. Pourquoi pas ? Si la recherche et l'affichage sont performants, il n'en va pas de même pour la saisie, laquelle exige que l'on retape entièrement une ligne en cas d'erreur (pas d'insertion... et il faut absolument repasser sur tous les caractères pour les valider ; sur 40 colonnes, ce n'est pas rédhibitoire !).

Vous constaterez que ce projet de répertoire constitue en fait une petite **boîte à idées.**

LIMITATIONS :

- Pas plus de 40 caractères par ligne.
- Pas plus de 100 lignes (actuellement).
- Le sous-programme de recherche ne transforme pas les capitales en minuscules et vice versa : pensez-y quand vous chercherez une lettre, un mot, un chiffre.

AVANTAGES :

- Programme et fichier sont chargés en même temps.
- Lancement par BRUN ALLO ou -ALLO.
- En cas de problème, un CALL 2049 vous remettra sur les rails.
- Vous pourrez de toutes façons sauver programme et fichier par "BSAVE ALLO, A$801,L4575".
- La touche ESCAPE abrège le listage (aussi bien en mode LISTE qu'en mode RECHERCHE).
- Les corrections sont très faciles (choisir l'option **2** ; ensuite, après avoir repéré la mauvaise ligne, taper **ESCAPE**, puis choisir l'option **1**, c'est-à-dire RECHERCHE, en donnant l'un des mots-clés de la ligne erronée... le reste coule de source...).

REMARQUES :

- Pendant le listage, on peut revenir à la ligne de menu en tapant ESCAPE.
- Lors d'une recherche, les lignes dans lesquelles figurent la chaîne recherchée s'affichent une à une, avec une pause permettant une éventuelle correction. Là encore, on peut abréger en pressant ESCAPE.
- Eviter, en mode EXEC, de démarrer en 80 colonnes (voir la petite routine en Basic à la fin du programme). Avec BRUN, c'est sans importance.
- Après un CALL 2049, un message d'erreur suivra probablement la sauvegarde. Ce sera sans importance.

DÉSASSEMBLAGE :

Le désassemblage du programme a été obtenu sur un Apple IIGS, d'où une certaine simplification dans le libellé des instructions. **Exemple** : **LDA £95** au lieu de **LDA £$95** sur un Apple *II* classique.

INITIALISATION :

- Si vous utilisez le programme de la disquette, aucun problème : il vous suffira de remplacer (CORRECTION) les deux lignes existantes par les vôtres.
- Pour remettre toute la partie fichier à 0, respectez le processus suivant :

- NEW	- CALL - 151 RETURN
- RENAME ALLO, ALLO0 (pour le cas où...)	- A00 : 0 RETURN - A01 < A00.19DF M
- BLOAD ALLO0	- BSAVE ALLO, A$801,L4575

Attention ! veillez à ne taper aucune ligne de Basic : elle bousculerait le début de la routine (installée à partir de $801). N'essayez pas non plus de lister ALLO... comme s'il s'agissait d'un programme en Basic.

PRÉPARATION

```
801 : A9 95       LDA £95      ┐ CTRL-U-CHR$ (21) - désactive la carte 80
803 : 20 ED FD    JSR FDED     ┘ colonnes (pas toujours efficace).
806 : 20 58 FC    JSR FC58       Home.
809 : A0 00       LDY £00      ┐
80B : B9 7E 09    LDA 097E,Y   │
80E : F0 06       BEQ 0816     │ Affichage de la ligne de menu
810 : 99 D0 07    STA 07D0,Y   │ sur la dernière ligne de l'écran.
813 : C8          INY          │
814 : D0 F5       BNE 080B     ┘
816 : 20 5A 09    JSR 095A       Vers PAUSE.
819 : C9 B0       CMP £B0        Est-ce le choix 0 - QUITTER ($B0) ?
81B : 90 F9       BCC 0816       Plus petit, c'est refusé.
81D : 8D F6 07    STA 07F6       On poque ce choix en bout de ligne.
820 : F0 0D       BEQ 082F       Si c'est 0, MEMORISATION.
822 : C9 B2       CMP £B2        Est-ce le choix 2 - LISTER ($B2) ?
824 : F0 19       BEQ 083F       Oui, on va donc à LISTE.
826 : 90 3D       BCC 0865       Plus petit, c'est le choix 1 - RECHERCHE.
828 : C9 B4       CMP £B4      ┐ Est-ce plus grand ou égal à 4 ($B4) ?
82A : B0 EA       BCS 0816     ┘ Oui, on refuse illico.
82C : 4C 18 09    JMP 0918       Sinon, direction AJOUT.
```

MÉMORISATION

```
82F : A0 00       LDY £00      ┐
831 : B9 A6 09    LDA 09A6,Y   │ La ligne de commande est envoyée dans le
834 : F0 06       BEQ 083C     │ buffer (BSAVE ALLO,A$801,L$11DF). On
836 : 99 00 02    STA 0200,Y   │ mémorise programme et fichier en bloc.
839 : C8          INY          │ ProDOS se charge de tout.
83A : D0 F5       BNE 0831     │
83C : 4C 03 BE    JMP BE03     ┘
```

LISTE

```
83F : A9 00       LDA £00      ┐
841 : 85 06       STA 06       │ L'adresse du fichier est mise en place
843 : A9 0A       LDA £0A      │ page zéro.
845 : 85 07       STA 07       ┘
847 : A2 15       LDX £15        X va servir de compteur pour les lignes (0 à 21).
849 : A0 00       LDY £00      ┐
84B : B1 06       LDA (06),Y   │
84D : F0 BA       BEQ 0809     │ Affichage d'une ligne.
84F : 20 ED FD    JSR FDED     │ Si on lit un 0, sortie immédiate : c'est la fin du
852 : C8          INY          │ fichier. On affiche 40 caractères (les fins de
853 : C0 28       CPY £28      │ lignes sont des espaces $A0).
855 : D0 F4       BNE 084B     ┘
857 : 20 72 09    JSR 0972       Vers ajustement des pointeurs.
85A : CA          DEX          ┐ Une ligne de moins. Quand on aura X = $FF,
85B : 10 EC       BPL          ┘ on ne bouclera plus !
85D : 20 5A 09    JSR 095A       Sous-programme PAUSE.
860 : 20 58 FC    JSR FC58       Home pour une nouvelle page.
863 : 80 E2       BRA 0847       Saut obligatoire.
```

(suite du listage page 16)

```
865 : 20 58 FC    JSR FC58      Home.
868 : A0 00       LDY £00       Y = 0, c'est plus prudent.
86A : B9 BF 09    LDA 09BF,Y    ┐
86D : F0 06       BEQ 0875      │ Affichage d'un petit texte
86F : 20 ED FD    JSR FDED      │ (VOUS RECHERCHEZ :).
872 : C8          INY           │
873 : D0 F5       BNE 086A      ┘
875 : 20 3A FF    JSR FF3A      Bell et son bip.
878 : 20 6F FD    JSR FD6F      GETLIN sans curseur.
87B : E0 00       CPX £00       ┐ La longueur de la chaîne à rechercher
87D : D0 03       BNE 0882      │ est dans X. Si c'est 0,
87F : 4C 06 08    JMP 0806      ┘ inutile de s'attarder ici !
882 : 86 FF       STX FF        X pour mémoire.
884 : BD FF 01    LDA 01FF,X    ┐
887 : 9D 9F 11    STA 119F,X    │ On mémorise le mot recherché, pour le
88A : CA          DEX           │ plaisir de l'avoir quelque part.
88B : D0 F7       BNE 0884      ┘
88D : A9 D8       LDA £D8       ┐
88F : 85 06       STA 06        │ Adresse de début de fichier,
891 : A9 09       LDA £09       │ mais moins $28.
893 : 85 07       STA 07        ┘
895 : A9 04       LDA £04       ┐ VTAB spécial.
897 : 20 5B FB    JSR FB5B      ┘
89A : 20 72 09    JSR 0972      Sous-programme d'AJUSTEMENT de l'adresse.
89D : 20 1A FC    JSR FC1A      Monte le curseur (c'est inutile dans cette version).
8A0 : A0 00       LDY £00       ┐ Y et X à zéro, comme on le voit.
8A2 : A2 00       LDX £00       ┘
8A4 : B1 06       LDA (06),Y    ┐ Lecture du caractère numéro 1 de la ligne.
8A6 : F0 6D       BEQ 0915      ┘ Si c'est 0, fin de fichier.
8A8 : BD A0 11    LDA 11A0,X    ┐ Lecture d'un caractère de la chaîne et com-
8AB : D1 06       CMP (06),Y    ┘ paraison avec celui de rang Y de la ligne.
8AD : D0 0C       BNE 08BB      Si ce n'est pas égal, vers INCY.
8AF : E8          INX           Sinon X = X + 1.
8B0 : E4 FF       CPX FF        ┐ Comparaison avec la longueur du mot
8B2 : B0 0E       BCS 08C2      ┘ recherché. Plus grand ou égal, c'est terminé.
8B4 : C8          INY           ┐ Incrémentation normale de Y.
8B5 : C0 28       CPY £28       │ S'il n'est pas égal à $28 (0 à 27),
8B7 : D0 EB       BNE 08A4      ┘ on continue.
8B9 : 80 DA       BRA 0895      Dans le cas contraire, voir une autre ligne.
8BB : C8          INY           ┐
8BC : C0 28       CPY £28       │ Ici, Y = Y + 1 aussi, mais on va remettre
8BE : 90 E2       BCC 08A2      │ X à 0 en $8A2.
8C0 : 80 D3       BRA 0895      ┘
8C2 : 20 62 FC    JSR FC62      CR retour chariot.
8C5 : A0 00       LDY £00       ┐
8C7 : B1 06       LDA (06),Y    │
8C9 : 20 ED FD    JSR FDED      │ Affichage de la ligne trouvée (peut-être Y
8CC : C8          INY           │ en a-t-il d'autres... on le verra plus loin).
8CD : C0 28       CPY £28       │
8CF : 90 F6       BCC 08C7      ┘
```

CORRECTION ?

```
8D1 :  A0 00      LDY £00
8D3 :  B9 D1 09   LDA 09D1,Y
8D6 :  F0 06      BEQ 08DE
8D8 :  99 50 06   STA 0650,Y
8DB :  C8         INY
8DC :  80 F5      BRA 08D3
8DE :  20 5A 09   JSR 095A
8E1 :  C9 83      CMP £83
8E3 :  D0 B0      BNE 0895
8E5 :  C6 25      DEC 25
8E7 :  20 22 FC   JSR FC22
8EA :  20 3A FF   JSR FF3A
8ED :  20 6F FD   JSR FD6F
8F0 :  8A         TXA
8F1 :  F0 1F      BEQ 0912
8F3 :  C9 29      CMP £29
8F5 :  90 02      BCC 08F9
8F7 :  A9 28      LDA £28
8F9 :  85 FF      STA FF
8FB :  A0 00      LDY £00
8FD :  B9 00 02   LDA 0200,Y
900 :  91 06      STA (06),Y
902 :  C8         INY
903 :  C4 FF      CPY FF
905 :  90 F6      BCC 08FD
907 :  A9 A0      LDA £A0
909 :  C0 28      CPY £28
90B :  F0 05      BEQ 0912
90D :  91 06      STA (06),Y
90F :  C8         INY
910 :  D0 F5      BNE 0907
912 :  4C 95 08   JMP 0895
915 :  4C 06 08   JMP 0806
```

Désire-t-on corriger cette ligne ?
(TAPEZ CTRL - C POUR CORRIGER : ?)

Sous-programme PAUSE (avec ESCAPE POSSIBLE).

Est-ce CTRL-C ? Non : ça repart pour une autre ligne éventuelle.

VTAB nous ramène au début de la ligne.

Bell (elle à l'habitude !).

La nouvelle ligne sera dans A...

Si la longueur est nulle on se fait un petit saut.

Si elle est plus grande que $28, on la raccourcit.

Mémo de cette longueur dans $FF.

Transfert du contenu du buffer dans le fichier.

Espace dans A.

Y est-il à $28 ? Si oui, c'est tout bon !

Sinon on complète la ligne avec des espaces.

Et on va voir si d'autres lignes existent.

Retour à la ligne de menu.

AJOUT

```
918 :  A9 00      LDA £00
91A :  85 06      STA 06
91C :  A9 0A      LDA £0A
91E :  85 07      STA 07
920 :  A2 64      LDX £64
922 :  A0 00      LDY £00
924 :  B1 06      LDA (06),Y
926 :  C9 00      CMP £00
928 :  F0 0F      BEQ 0939
92A :  C9 A0      CMP £A0
92C :  F0 0B      BEQ 0939
92E :  CA         DEX
92F :  F0 05      BEQ 0936
931 :  20 72 09   JSR 0972
934 :  80 EE      BRA 0924
936 :  4C 06 08   JMP 0806
939 :  20 58 FC   JSR FC58
```

Initialisation des pointeurs du fichier.

Compteur pour 100 lignes.

On recherche une ligne vide, n'importe laquelle. Si on trouve 00 (fin de fichier) ou $A0 au début d'une ligne, c'est qu'elle est vide. **Moralité** : Ne pas commencer une donnée par un espace !
Sitôt trouvée, on va plus loin.
Dans le cas contraire, retour sans gloire.

Vers sous-programme pour ajustement des pointeurs.

Saut obligé pour une nouvelle recherche.

Retour ligne de menu.

LÀ, C'EST BON.

(suite page 18)

```
93C :  B9  EF  09     LDA  09EF,Y
93F :  F0  06         BEQ  0947
941 :  20  ED  FD     JSR  FDED
944 :  C8             INY
945 :  80  F5         BRA  093C
947 :  A0  28         LDY  #28
949 :  A9  AE         LDA  #AE
94B :  99  FF  05     STA  05FF,Y
94E :  88             DEY
94F :  D0  FA         BNE  094B
951 :  84  24         STY  24
953 :  A9  04         LDA  #04
955 :  20  5B  FB     JSR  FB5B
958 :  80  90         BRA  08EA
```

Message : NOUVELLE LIGNE :

Ligne de points pour éclairer les ignorants.

Y est à 0, le voilà dans CH.

VTAB spécial.

Et on profite de la routine de correction.

PAUSE

```
95A :  20  3A  FF     JSR  FF3A
95D :  AD  00  C0     LDA  C000
960 :  10  FB         BPL  095D
962 :  2C  10  C0     BIT  C010
965 :  C9  9B         CMP  #9B
967 :  D0  08         BNE  0971
969 :  68             PLA
96A :  68             PLA
96B :  A9  08         LDA  #08
96D :  48             PHA
96E :  A9  05         LDA  #05
970 :  48             PHA
971 :  60             RTS
```

BELL et son bip.

On attend qu'une touche soit pressée.

Clavier remis à zéro.

Si ce n'est pas ESCAPE, saut.

Sinon, adresse destinataire moins un sur la pile.

Retour... ou $806.

AJUSTEMENT DES POINTEURS

```
972 :  A5  06         LDA  06
974 :  18             CLC
975 :  69  28         ADC  #28
977 :  85  06         STA  06
979 :  90  02         BCC  097D
97B :  E6  07         INC  07
97D :  60             RTS
```

L'adresse $06-07 est augmentée de $28 (40). Le pointeur $07 n'est incrémenté que lorsque la retenue est à 1.

TEXTES

```
97E :  30 A0-O
980 :  D1 D5 C9 D4 D4 C5 D2 A0 31 A0 D2 C5 C3 C8 C5 D2 -QUITTER 1 RECHER
990 :  C3 C8 C5 A0 32 A0 CC C9 D3 D4 C5 A0 33 A0 C1 CA -CHE 2 LISTE 3 AJ
9A0 :  CF D5 D4 A0 3F 00 C2 D3 C1 D6 C5 A0 C1 CC CC CF -OUT ?.BSAVE ALLO
9B0 :  AC C1 A4 B8 B0 B1 AC CC A4 B1 B1 C4 C6 8D 00 D6 - ,A$801,L$11DF..V
9C0 :  CF D5 D3 A0 D2 C5 C3 C8 C5 D2 C3 C8 C5 DA BA A0 -OUS RECHERCHEZ:
9D0 :  00 D4 C1 D0 C5 DA A0 03 14 12 0C AD 03 A0 D0 CF - .TAPEZ ....-. PO
9E0 :  D5 D2 A0 C3 CF D2 D2 C9 C7 C5 D2 BA A0 3F 00 CE -UR CORRIGER: ?.N
9F0 :  CF D5 D6 C5 CC CC C5 A0 CC C9 C7 CE C5 A0 BA 00 -OUVELLE LIGNE :.
A00 :  D4- T
```

ALFEX

```
10 D$ = CHR$ (4)
20 PRINT D$"OPEN ALLO.EXEC"
30 PRINT D$"WRITE ALLO.EXEC"
35 PRINT "PRINT CHR$ (21):HOME"
40 PRINT "PREFIX/R4/ALL"
45 PRINT "-ALLO"
50 PRINT "-MENU,S6"
55 PRINT D$"CLOSE"
```

Pour lancer ALLO

Pour lancer **ALLO** à partir d'un menu de disquette, on peut passer par une commande EXEC qui va successivement :

- Déconnecter la carte 80 colonnes (il vaut mieux le faire à partir du Basic) ;
- Vider l'écran ;
- Installer le préfixe du sous-catalogue éventuel (autre syntaxe, pour un slot différent : "PREFIX,S5", par exemple) ;
- BRUN **ALLO** ;
- Retour au menu (ou à tout autre programme) en fin de session.

Un **RUN ALFEX** (c'est le nom de cette mini-routine) va créer, une fois pour toute, la commande **ALLO.EXEC**, laquelle sera activée par un :

- ALLO.EXEC (sous DOS 3.3, on taperait **EXEC ALLO.EXEC**).

BON À SAVOIR :

Si, après avoir consulté **ALLO**, vous désirez gagner du temps, vous pouvez en sortir en tapant un **CTRL-RESET** qui (la commande EXEC étant toujours active) vous renverra directement au MENU (cas présent) ou à tout autre programme de votre choix... mais sans mémorisation préalable d'**ALLO**.

INITIALISATION

(lire page 14)

Pour initialiser un fichier **ALLO**, vous pouvez aussi utiliser cette routine en Basic. Sur l'Apple IIGS, ne pas oublier la modification indiquée par la REM de la ligne 15.

ALCLR

```
10 TEXT : NORMAL : HOME
15 REM Sur l'Apple IIGS, faire précéder les adresses par 00, comme ceci :A$ =
   "00/A00:0 N 00/A01 < A00.19DFM D823G"
20 A$ = "A00:0 N A01 < A00.19DFM D823G"
30 FOR I = 1 TO LEN (A$): POKE 511 + I, ASC ( MID$ (A$,I,1)) + 128: NEXT :
   POKE 72,0: CALL - 144: PRINT
```

■

ALIGN.DR40

OBJET :

Afficher une chaîne avec alignement à droite (sur un écran de 40 colonnes).

REMARQUE :

On écrit directement le code de chaque caractère dans la page TEXT et il n'y a donc pas de problème de scrolling (déroulement des lignes d'écran vers le haut) lorsque l'affichage concerne la ligne 24. VTAB n'est pas modifié lors de l'exécution de la routine LM.

```
100 TEXT : PRINT CHR$ (21)
105 FOR I = 768 TO 813: READ R: POKE I,R:
    NEXT
110 DATA 32,190,222,32,227,223,160,2,177,
    131,153,24,0,136,16,248,32,245,230,138,
    32,71,248,169,40,56,229,24,24,101,38,
    133,38,164,24,136,177,25,9,128,145,38,
    136,16,247,96
115 VTAB 1: CALL - 958
120 INVERSE : PRINT " UNE CHAINE DE 40
    CARACTERES AU MAXIMUM ": NOR
    MAL
125 VTAB 3: INPUT "";C$: IF C$ = "" THEN
    160
130 IF LEN (C$) > 40 THEN 115
135 VTAB 4: INPUT "Sur quelle ligne l'affi
    cher (avec 8 ESP. alignement à droi
    te ? ";L$: IF L$ = "" THEN 135
140 L = VAL (L$) - 1: IF L < 0 OR L > 23
    THEN 135
145 CALL 768,C$,L
150 GOSUB 180: IF PEEK (49152) = 27 THEN
    160
155 HOME : FOR I = 0 TO 23: CALL 768,C$,I:
    NEXT : GOSUB 180: GOTO 115
160 A$ = "(M)ENU DE DISQUETTE (A)PPLE
    SOFT      ":L = 23: CALL 768,A$,L
165 GOSUB 180: IF PEEK (49152) = 77 THEN
    PRINT CHR$ (4)"RUN MENU"
170 IF PEEK (49152) = 65 THEN HOME : END
175 : GOTO 165
180 CALL - 198: POKE 49168,0: WAIT
    49152,128: POKE 49168,0: RETURN
```

PARAMÈTRES

```
300 :  20  BE  DE      JSR  $DEBE
303 :  20  E3  DF      JSR  $DFE3
306 :  A0  02          LDY  £$02
308 :  B1  83          LDA  ($83),Y
30A :  99  18  00      STA  $0018,Y
30D :  88              DEY
30E :  10  F8          BPL  $0308
310 :  20  F5  E6      JSR  $E6F5
313 :  8A              TXA
314 :  20  47  F8      JSR  $F847
```

CHKCOM teste la présence de la virgule et PTCHARGET recherche la variable par son nom.

On va ainsi retrouver la longueur de cette variable dans $18 et son adresse en $19-1A.

GETBYTC, puis GETBYT et CONINT. Un caractère est sauté, l'expression évaluée et le résultat récupéré dans X. On le passe dans A pour que GBASCALC donne l'adresse de la ligne.

HTAB

```
317 :  A9  28          LDA  £$28
319 :  38              SEC
31A :  E5  18          SBC  $18
31C :  18              CLC
31D :  65  26          ADC  $26
31F :  85  26          STA  $26
```

Longueur de ligne moins longueur du mot = position dans la ligne (HTAB en Basic). Comme nous disposons (dans $26-27), grâce à GBASCALC (voir ci-dessus), de l'adresse de cette ligne, il suffit d'en modifier la partie basse en lui additionnant HTAB.

AFFICHAGE

```
321 :  A4  18          LDY  $18
323 :  88              DEY
324 :  B1  19          LDA  ($19),Y
326 :  09  80          ORA  £$80
328 :  91  26          STA  ($26),Y
32A :  88              DEY
32B :  10  F7          BPL  $0324
32D :  60              RTS
```

Longueur du mot (-1) dans Y.

Lecture d'un caractère
BIT 7 à 1 par ORA 10000000.
Affichage du caractère (c'est un poke).
Y = Y - 1.
On continue jusqu'à Y = 0 inclus.
RETOUR AU BASIC.

Pour utiliser la routine sans DATA, sauvez-la sous le nom de votre choix par :

BSAVE MONPROGR, A$300,L$2E

UTILISATION :

Elle peut être relogée à n'importe quelle adresse disponible. Vous y aurez accès par un **CALL ADR, V$, L**
où ADR = adresse, V$ = chaîne à afficher et L = VTAB - 1.

ÉQUIVALENT BASIC (ALIGN.BAS0)

AVEC SCROLLING

```
10 TEXT : PRINT CHR$ (21): HOME
15 A$ = "CECI EST UN ESSAI":V = 12
20 HTAB 41 - ( LEN (A$)): VTAB 12: PRINT A$
```

SANS SCROLLING

```
25 V = 24
30 VTAB V
35 D = ( PEEK (42) + PEEK (43) * 256) + (39 - LEN (A$))
40 FOR I = 1 TO LEN (A$): POKE I + D, ASC ( MID$ (A$,I,1)) + 128: NEXT
45 VTAB 8: PRINT "(M)ENU DE DISQUETTE ";: GET R$: IF R$ = "M" THEN PRINT CHR$
   (4)"RUN MENU"
```

■

ALIGN.DR80

Programme similaire au précédent, mais avec un affichage en 80 colonnes, obtenu grâce à une action conjuguée de COUT ($FDED) et d'un POKE (le dernier caractère de la ligne... ou de la variable, ce qui revient au même).

```
100  TEXT : PRINT CHR$ (4)"PR£3"
105  FOR I = 768 TO 825: READ R: POKE I,R:  NEXT
110  DATA     32,190,222,32,227,223,160,2,177,131,153,24,0,136,16,248,32,245,230,
     138,32,91,251,138,32,71,248,169,80,56,229
115  DATA     24,141,123,5,198,24,160,0,177,25,9,128,196,24,240,6,32,237,253,200,
     128,242,160,39,145,38,96
120  PRINT : VTAB 1: CALL - 958
125  INVERSE : PRINT "CHAINE DE 80 CARACTERES MAXIMUM  ": NORMAL
130  VTAB 3: INPUT "";C$: IF C$ = "" THEN  165
135  IF LEN (C$) > 80 THEN 120
140  VTAB 4: INPUT "Sur quelle ligne l'afficher (avec alignement à droite ? ";
     L$: IF L$ = "" THEN 140
145  L = VAL (L$) - 1: IF L < 0 OR L > 23  THEN 140
150  CALL 768,C$,L
155  GOSUB 185: IF PEEK (49152) = 27 THEN  165
160  HOME : FOR I = 0 TO 23: CALL 768,C$,I:  NEXT : GOSUB 185: GOTO 120
165  A$ = "(M)ENU DE DISQUETTE (A)PPLESOFT        ":L = 23: CALL 768,A$,L
170  GOSUB 185: IF PEEK (49152) = 77 THEN  PRINT CHR$ (4)"RUN MENU"
175  IF PEEK (49152) = 65 THEN HOME : END
180  GOTO 170
185  CALL - 198: POKE 49168,0: WAIT 49152,128: POKE 49168,0: RETURN
```

```
300 :   20  BE  DE      JSR  $DEBE
303 :   20  E3  DF      JSR  $DFE3
306 :   A0  02          LDY  £$02
308 :   B1  83          LDA  ($83),Y
30A :   99  18  00      STA  $0018,Y
30D :   88              DEY
30E :   10  F8          BPL  $0308
```

On stocke la longueur de la variable en $18 et son adresse en $19-1A.

```
310 : 20 F5 E6     JSR $E6F5
313 : 8A           TXA
314 : 20 5B FB     JSR $FB5B
317 : 8A           TXA
318 : 20 47 F8     JSR $F847
31B : A9 50        LDA £$50
31D : 38           SEC
31E : E5 18        SBC $18
320 : 8D 7B 05     STA $057B
323 : C6 18        DEC $18
325 : A0 00        LDY £$00
327 : B1 19        LDA ($19),Y
329 : 09 80        ORA £$80
32B : C4 18        CPY $18
32D : F0 06        BEQ $0335
32F : 20 ED FD     JSR $FDED
332 : C8           INY
333 : 80 F2        BRA $0327
335 : A0 27        LDY £$27
337 : 91 26        STA ($26),Y
339 : 60           RTS
```

Le numéro de ligne (trouvé dans X) passe dans A pour que TABV ajuste la position du curseur.

X de nouveau dans A pour que GBASCALC place l'adresse de la ligne en $26-27.

Position horizontale du curseur (H) = longueur de ligne ($50) - longueur du texte à afficher.

POKE 1403,H... du Basic.

Longueur décrémentée de 1.

Boucle initialisée.

Lecture du caractère.

Bit 7 à 1 avec ORA 10000000.

Si Y est égal à la longueur, c'est le dernier caractère...

Sinon COUT affiche le caractère.

Boucle incrémentée et saut obligé pour continuer.

Le dernier caractère (ou le seul quand la longueur est 1) est poquée en bout de ligne.

RETOUR AU BASIC.

BSAVE MONPROGRAMME,A$300,L$3A

Attention ! contrairement à **ALIGN.DR40**, cette routine modifie la position verticale du curseur. En tenir compte quand l'affichage est demandé en VTAB 24. Prévoir (par exemple) un **VTAB 22 : PRINT**, au retour.

ALIGN.MIXTE

```
100 TEXT : PRINT CHR$ (4)"PR£3"
105 FOR I = 768 TO 832: READ R: POKE I,R: NEXT
110 DATA 32,190,222,32,227,223,160,2,177,131,153,24,0,136,16,248,32,245,230,
    138,32,91,251,138,32,71,248,169,40,44,31,192,16
115 DATA 2,169,80,56,229,24,141,123,5,198,24,160,0,177,25,9,128,196,24,240,
    6,32,237,253,200,128,242,160,39,145,38,96
120 PRINT : VTAB 1: CALL - 958: PRINT "(4)0 COLONNES (8)0 COLONNES ? ";:
    GET R$: IF R$ = "4" THEN PRINT CHR$ (17)
125 IF R$ = "8" THEN PRINT CHR$ (18)
130 IF R$ < > "4" AND R$ < > "8" THEN 120
135 PRINT : VTAB 1
```

(suite page 24)

```
140 INVERSE : PRINT "CHAINE DE "R$"0 CARACTERES MAXIMUM": NORMAL
145 VTAB 3: INPUT "";C$: IF C$ = "" THEN 180
150 IF LEN (C$) > VAL (R$) * 10 THEN 120
155 VTAB 4: INPUT "Sur quelle ligne l'afficher (avec alignement à droite ? ";L$: IF L$ = "" THEN 155
160 L = VAL (L$) - 1: IF L < 0 OR L > 23 THEN 155
165 CALL 768,C$,L
170 GOSUB 200: IF PEEK (49152) = 27 THEN 180
175 HOME : FOR I = 0 TO 23: CALL 768,C$,I: NEXT : GOSUB 200: GOTO 120
180 A$ = "(M)ENU DE DISQUETTE (A)PPLESOFT ": L = 23: CALL 768,A$,L
185 GOSUB 200: IF PEEK (49152) = 77 THEN PRINT CHR$ (4)"RUN MENU"
190 IF PEEK (49152) = 65 THEN HOME : END
195 GOTO 185
200 CALL - 198: POKE 49168,0: WAIT 49152,128: POKE 49168,0: RETURN
```

Finalement, il est facile d'obtenir une routine mixte, fonctionnant aussi bien en 40 qu'en 80 colonnes. La voici ! C'est rigoureusement la même que la précédente, à quelques octets près.

```
300 :  20 BE DE    JSR  $DEBE
303 :  20 E3 DF    JSR  $DFE3
306 :  A0 02       LDY  £$02
308 :  B1 83       LDA  ($83),Y
30A :  99 18 00    STA  $0018,Y
30D :  88          DEY
30E :  10 F8       BPL  $0308
310 :  20 F5 E6    JSR  $E6F5
313 :  8A          TXA
314 :  20 5B FB    JSR  $FB5B
317 :  8A          TXA
318 :  20 47 F8    JSR  $F847
31B :  A9 28       LDA  £$28
31D :  2C 1F C0    BIT  $C01F
320 :  10 02       BPL  $0324
322 :  A9 50       LDA  £$50
324 :  38          SEC
325 :  E5 18       SBC  $18
327 :  8D 7B 05    STA  $057B
32A :  C6 18       DEC  $18
32C :  A0 00       LDY  £$00
32E :  B1 19       LDA  ($19),Y
330 :  09 80       ORA  £$80
332 :  C4 18       CPY  $18
334 :  F0 06       BEQ  $033C
336 :  20 ED FD    JSR  $FDED
339 :  C8          INY
33A :  80 F2       BRA  $032E
33C :  A0 27       LDY  £$27
33E :  91 26       STA  ($26),Y
340 :  60          RTS
```

Supposons que nous sommes en 40 colonnes. Si c'est vrai, allons directement vers la soustraction. Sinon, A change de valeur pour 80 colonnes.

ÉQUIVALENT BASIC EN 80 COLONNES (ALIGN.BAS1)

```
10 TEXT :D$ = CHR$ (4): PRINT D$"PR£3": PRINT : HOME
15 A$ = "CECI EST SEULEMENT UN ESSAI"
20 V = 24: VTAB V
25 D = ( PEEK (40) + PEEK (41) * 256) + (39 — LEN (A$))
30 FOR I = 1 TO LEN (A$): POKE I + D, ASC ( MID$ (A$,I,1)) + 128: NEXT
35 GOSUB 65
40 HOME : VTAB 24: POKE 1403,80 - LEN (A$): PRINT LEFT$ (A$, LEN (A$) - 1);: POKE D + LEN (A$), ASC ( RIGHT$ (A$,1)) + 128
45 VTAB 1: PRINT "": GOSUB 65
50 VTAB 24: PRINT "(M)ENU DE DISQUETTE ";: GET R$: VTAB 22: PRINT ""
55 IF R$ = "M" THEN PRINT D$"RUN /R4/MENU"
60 HOME : END
65 CALL - 198: POKE 49168,0: WAIT 49152,128: POKE 49168,0: RETURN
```

JEU.ALTERNÉ

Il est facile, en utilisant la petite astuce de programmation ci-après, d'aller alternativement activer deux sous-programmes. Mais on peut envisager d'autres possibilités en utilisant des valeurs différentes pour le compteur. Essayez par exemple $CA... Pour obtenir 8 coups, il est indispensable que le bit 7 du compteur soit à 1.

```
300 :  A9 AA      LDA  £$AA     10101010 dans A (compteur).
302 :  4A         LSR           Un bit sort à droite et tombe dans la retenue.
303 :  48         PHA           Contenu de A sur la pile.
304 :  90 04      BCC  $030A    Si la retenue est à 0, saut.
306 :  A9 31      LDA  £$31     ┐ Sinon 1 (inverse) dans A pour COUT...
308 :  D0 02      BNE  $030C    ┘ et saut.
30A :  A9 30      LDA  £$30     0 dans A (en mode inverse).
30C :  20 ED FD   JSR  $FDED    COUT affiche le contenu de A.
30F :  68         PLA           Récupération du compteur sur la pile.
310 :  D0 F0      BNE  $0302    Si A différent de 0, on continue.
312 :  60         RTS           Terminé.
```

```
* 300G
01010101
```

TEST n°1

```
300 :  A9 AA      LDA  £$AA     ┐
302 :  4A         LSR           │
303 :  48         PHA           │
304 :  B0 04      BCS  $030A    │
306 :  A9 31      LDA  £$31     │
308 :  D0 02      BNE  $030C    │
30A :  A9 30      LDA  £$30     │
30C :  20 ED FD   JSR  $FDED    │
30F :  68         PLA           │
310 :  D0 F0      BNE  $0302    │
312 :  60         RTS           ┘
```

Même processus, mais on utilise BCS en $304 au lieu de BCC. L'ordre est inversé. Le **01010101** du premier exemple devient évidemment **10101010**.

```
* 300G
10101010
```

TEST n°2

■

COPIE.HGR

Cette routine permet de recopier une image graphique sur les imprimantes Apple (DMP, IW et IWII). Elle n'autorise pas le double format en hauteur. Toutefois, un **PRINT CHR$ (14)** précédant l'impression fournira une image déformée, en double largeur. On n'oubliera pas, le cas échéant, de modifier la valeur de la marge.

Il est possible, en utilisant certaines options de l'imprimante, d'obtenir des reproductions plus "pointues" (voir nos exemples).

Attention ! Si vous mettez votre imprimante en double largeur, glissez un **"PRINT CHR$ (15)"** dans votre programme en Basic pour revenir à la largeur normale... ou éteignez l'imprimante après chaque impression.

IMAGEWRITER IMAGEWRITER 2 DMP APPLE

HGR ou HGR2

PAGE 1 : POKE 7,1
PAGE 2 : POKE 7,2

IMPRESSION

Noir sur blanc : POKE 6,1
Blanc dans noir : POKE 6,0
POKE 6,0 convient pour les photos et POKE 6,1 pour les graphiques.

MARGE

La marge est fixée à 10 (L010). Pour la modifier, taper : **M$ = "L020"** (par exemple), puis **X = 0 : FOR I = 956 TO 953 STEP - 1: X = X + 1: POKE I, ASC (MID$(M$, X, 1)) : NEXT**

HAUTEUR

POKE 253,0 = ligne 1
POKE 254,24 = ligne 24

CARACTÈRE

Le caractère actuel est **n** (camemberts ronds). Pour le changer : POKE 951 ASC(N ou E... par exemple).

QUALITÉ

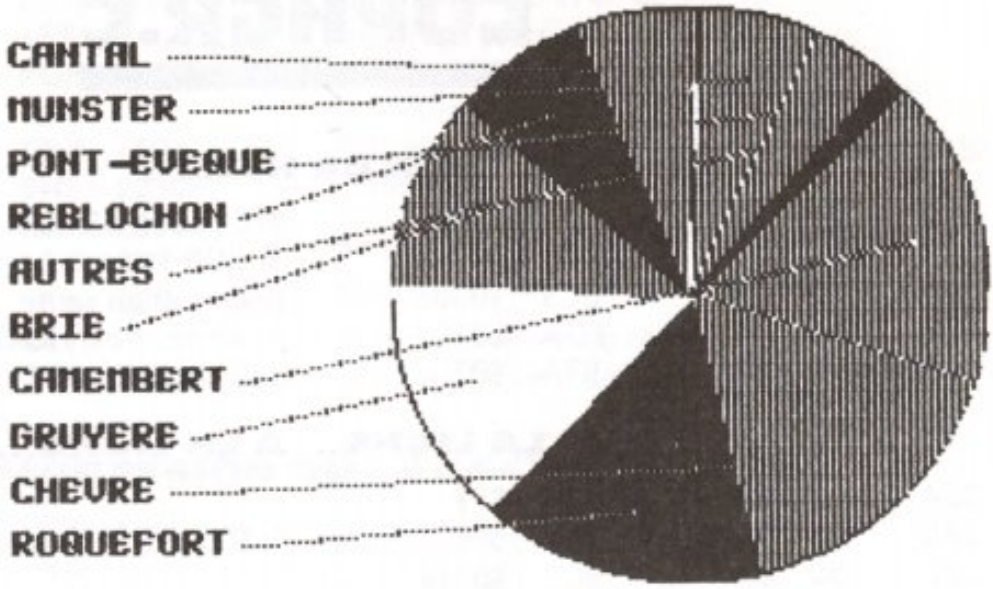
L'impression peut être améliorée en mettant préalablement l'imprimante en gras : PRINT CHR$(27) ; "!".

```
10 HGR : TEXT : HOME : D$ = CHR$ (4): PRINT
   D$"PR£3": PRINT : PRINT D$ "PREFIX/R4 /COPIE
   .HGR"
15 PRINT D$"BLOAD COPHGR.C": REM A$300
16 GOTO 19: REM A supprimer pour d'éventuelles
   utilisations de la ligne 17
17 POKE 951, ASC ("Q"):M$ = "L010":X = 0: FOR I
   = 956 TO 953 STEP -1: X = X + 1: POKE I, ASC
   ( MID$ (M$,X,1)): NEXT: REM Option double lar
   geur nécessaire avec les caractères "q" et "Q"
19 PRINT D$"PR£1": PRINT CHR$ (15): PRINT D$
   "PR£0": REM CHR$ (14) pour double largeur
20 VTAB 22: CALL - 868: INPUT "TITRE DE L'IMAGE
   (COPEX PAR DEFAUT - C POUR CADRE
   D'ESSAI)  ";T$: IF T$ = "" THEN T$ = "COPEX"
25 IF T$ = "C" THEN HCOLOR= 3: POKE 230,32:
   HPLOT 0,0 TO 279,0 TO 279,191 TO 0,191 TO
   0,0: GOTO 35
30 PRINT D$"BLOAD";T$;",A$2000"
35 POKE 6,1: POKE 7,1: POKE 253,0: POKE 254,24
40 VTAB 22: CALL - 868: PRINT "(R)APIDE (Q)UALITE
   ";: GET R$: PRINT
45 C$ = CHR$ (33): REM Point d'exclamation
50 IF R$ = "R" OR R$ = "r" THEN C$ = CHR$ (34):
   REM Guillemets
55 GOSUB 95
60 CALL 768: PRINT : REM Ajouter un (PRINT
   D$"PR£3": PRINT) sur les anciens Apple£60
   CALL 768: PRINT: REM Ajouter un (PRINT
   D$"PR£3":PRINT) sur les anciens Apple
65 VTAB 22: PRINT "(E)NCORE (M)ENU DE DIS
   QUETTE (A)PPLESOFT ";: GET R$: PRINT
70 S = ASC (R$): IF S > 77 THEN R$ = STR$ (S - 64)
75 IF R$ = "M" THEN PRINT D$"RUN /R4/MENU"
80 IF R$ = "E" THEN 20
85 IF R$ < > "A" THEN 65
90 HOME : END
95 PRINT D$"PR£1": PRINT CHR$ (27);C$: PRINT
   D$"PR£0": RETURN
```

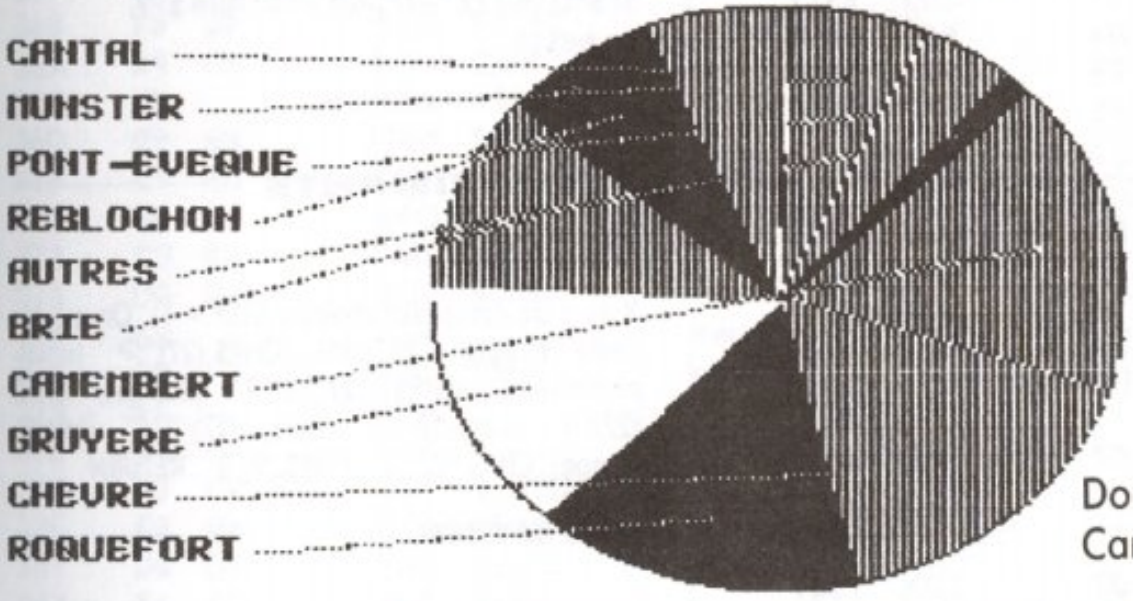
LES FROMAGES PREFERES DES FRANCAIS
D'APRES UNE ENQUETE DE LA SOFRES

OPTION QUALITÉ

Sans aucune modification des paramètres



LES FROMAGES PREFERES DES FRANCAIS
D'APRES UNE ENQUETE DE LA SOFRES



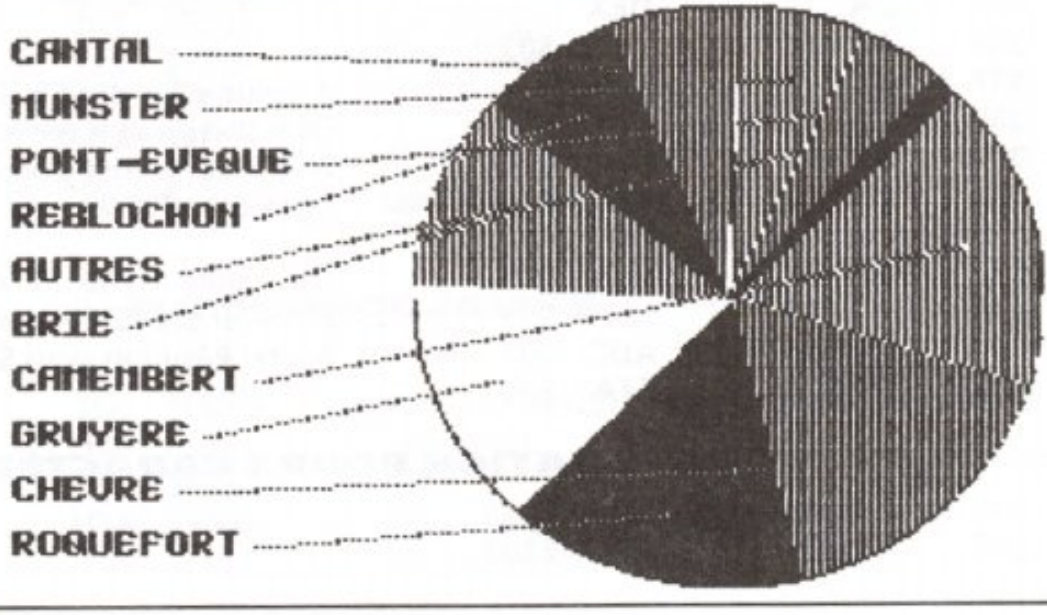
OPTION RAPIDE

Double largeur
Caractère "q"

LES FROMAGES PREFERES DES FRANCAIS
D'APRES UNE ENQUETE DE LA SOFRES

OPTION RAPIDE

Double largeur
Caractère "Q"



COPHGR.C

HGR ou HGR2 ?

```
300 :  A9  1C      LDA  £$1C
302 :  66  07      ROR  $07
304 :  B0  02      BCS  $0308
306 :  A9  3C      LDA  £$3C
308 :  85  07      STA  $07
```

$1C pour HGR dans A.
Si ROR fait tomber 1 dans la retenue, c'est page 1 et on saute...
... sinon, c'est HGR2.
Stockage dans $07.

DE LA LIGNE... À LA LIGNE...

```
30A :  A5  FE      LDA  $FE
30C :  C5  FD      CMP  $FD
30E :  B0  04      BCS  $0314
310 :  C6  FD      DEC  $FD
312 :  D0  F8      BNE  $030C
314 :  C9  19      CMP  £$19
316 :  90  04      BCC  $031C
318 :  C6  FE      DEC  $FE
31A :  D0  EE      BNE  $030A
```

Si $FD plus petit que $FE, saut.

Sinon $FD est décrémenté.

Si $FD (ou FE) est plus petit que $19, ça passe...

Sinon tout est à revoir.

INITIALISATION DE L'IMPRIMANTE

```
31C :  A9  01      LDA  £$01
31E :  20  95 FE   JSR  $FE95
321 :  A2  17      LDX  £$17
323 :  BD  AE 03   LDA  $03AE,X
326 :  20  ED FD   JSR  $FDED
329 :  CA          DEX
32A :  10  F7      BPL  $0323
```

PR£1 du Basic.

X sert de compteur (boucle STEP - 1). On envoie CHR$ (9)"80N" - RETURN - CHR$ (27)" > " (unidirectionnel), CHR$ (27)"L010" (marge), CHR$ (27)"n" (caractère), CHR$ (27)"T16" (interlignage), CHR$ (27)"Z", CHR$ (9)"Z", RETURN

BON POUR UNE LIGNE

```
32C :  A2  07      LDX  £$07
32E :  BD  C6 03   LDA  $03C6,X
331 :  20  ED FD   JSR  $FDED
334 :  CA          DEX
335 :  10  F7      BPL  $032E
337 :  E8          INX
338 :  86  FF      STX  $FF
33A :  A5  FD      LDA  $FD
33C :  20  C1 FB   JSR  $FBC1
33F :  65  FF      ADC  $FF
341 :  85  08      STA  $08
343 :  A5  29      LDA  $29
345 :  65  07      ADC  $07
347 :  85  09      STA  $09
```

"G0280" envoyé à l'imprimante (qui attend une ligne graphique).

X remis à 0 (il était à $FF).
On le stocke dans le pointeur $FF (longueur de ligne).
BASCALC fournit l'adresse de la première ligne dans $28-29.

On ajoute HTAB (lu dans FF) à la partie basse et $1C/$3C (selon la page) à la partie haute. Résultats dans $8-9.

PRÉPARATION POUR 1 CARACTÈRE

```
349 :  A0  08      LDY  £$08
34B :  A2  00      LDX  £$00
```

Y = 8 et X = 0

```
34D : A1 08     LDA ($08,X)   Lecture de l'octet en $8-9 + X.
34F : 4A        LSR           Décalage à droite.
350 : 76 18     ROR $18,X     La retenue entre à gauche.
352 : E8        INX           X = X + 1.
353 : E0 07     CPX £$07      ┐ S'il n'est pas égal à 7, on continue la
355 : D0 F8     BNE $034F     ┘ rotation.
357 : A9 03     LDA £$03      ┐
359 : 65 09     ADC $09       │ Sinon on majore la partie haute de 04 pour
35B : 85 09     STA $09       ┘ passer à la ligne suivante (+1024).
35D : 88        DEY           ┐ Y décrémenté et ça repart
35E : D0 EB     BNE $034B     ┘ si Y est différent de 0.
```

IMPRESSION DU CARACTÈRE

```
360 : A2 FF     LDX £$FF      ┐ Autrement dit X = 0.
362 : E8        INX           ┘
363 : B5 18     LDA $18,X     Lecture.
365 : A4 06     LDY $06       ┐ Mode (inverse ou normal).
367 : 88        DEY           ┘
368 : F0 02     BEQ $036C     Si 0, pas de EOR.
36A : 49 FF     EOR £$FF      Sinon on inverse.
36C : 48        PHA           A empilé.
36D : 09 80     ORA £$80      ┐
36F : C9 8D     CMP £$8D      │ Filtrage de $0D, $8D, $1A et $9A (CTRL-13 et
371 : F0 04     BEQ $0377     │ CTRL-Z). Ce n'est pas un procédé très ortho-
373 : C9 9A     CMP £$9A      │ doxe, mais qui donne des résultats corrects.
375 : D0 04     BNE $037B     ┘
377 : 68        PLA           ┐ Récupération sur la pile et petite transfor-
378 : 29 F7     AND £$F7      │ mation. Si caractère "dangereux". Il est de
37A : 48        PHA           ┘ nouveau empilé.
37B : 68        PLA           ┐ Impression correcte dans tous les cas, GS
37C : 20 ED FD  JSR $FDED     ┘ compris.
37F : E0 06     CPX £$06      X est-il égal à 6 ?
381 : D0 DF     BNE $0362     Non, on continue.
383 : E6 FF     INC $FF       ┐ + une tabulation.
385 : A5 FF     LDA $FF       ┘
387 : C9 28     CMP £$28      ┐ Si on n'est pas à $28, on repart pour un autre
389 : D0 AF     BNE $033A     ┘ caractère.
38B : 20 8E FD  JSR $FD8E     Retour chariot.
38E : E6 FD     INC $FD       + une ligne.
390 : A5 FD     LDA $FD       ┐
392 : C5 FE     CMP $FE       │ Si FD différent de FE, bon pour une autre
394 : 90 96     BCC $032C     ┘ ligne graphique.
```

FIN DE PROGRAMME

```
396 : A9 1B     LDA £$1B      ┐
398 : 20 ED FD  JSR $FDED     │ PRINT CHR$ (27)"c" (restaure les instructions
39B : A9 63     LDA £$63      │ standard).
39D : 20 ED FD  JSR $FDED     ┘
3A0 : A9 03     LDA £$03      ┐ PR£3 : ne suffit pas
3A2 : 20 95 FE  JSR $FE95     ┘ sur les anciens Apple.
```

(suite page 30)

```
3A5 :  A9  F1      LDA  £$F1
3A7 :  85  36      STA  $36
3A9 :  A9  B7      LDA  £$B7
34B :  85  37      STA  $37
34D :  60          RTS
```

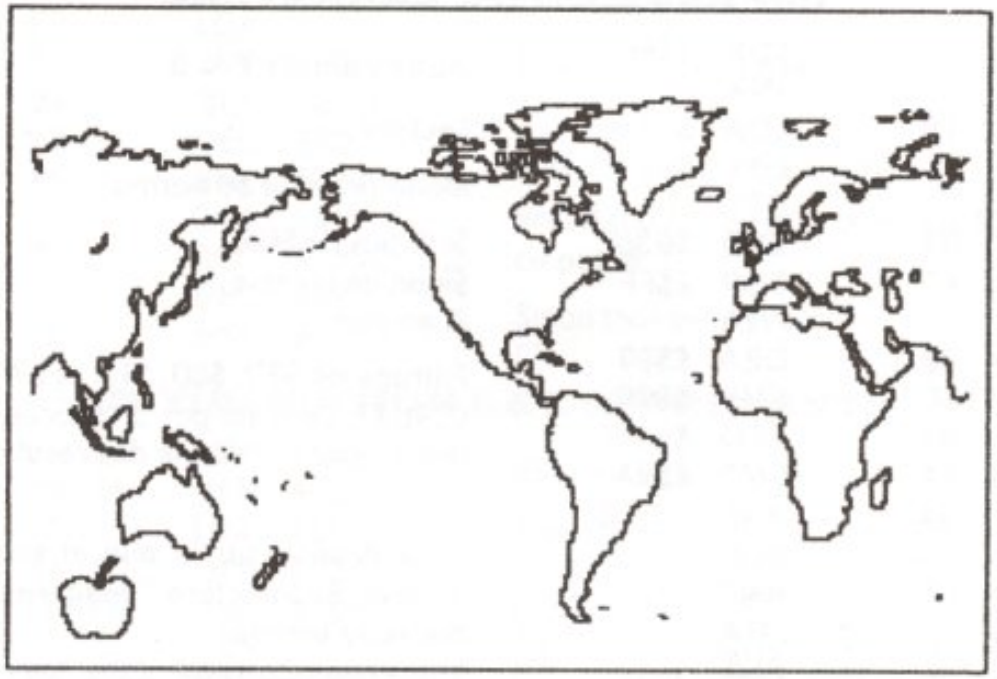
Routine de sortie de caractères restaurée (tout cela est pour ProDOS).

Retour.

PARAMÈTRES IMPRIMANTE

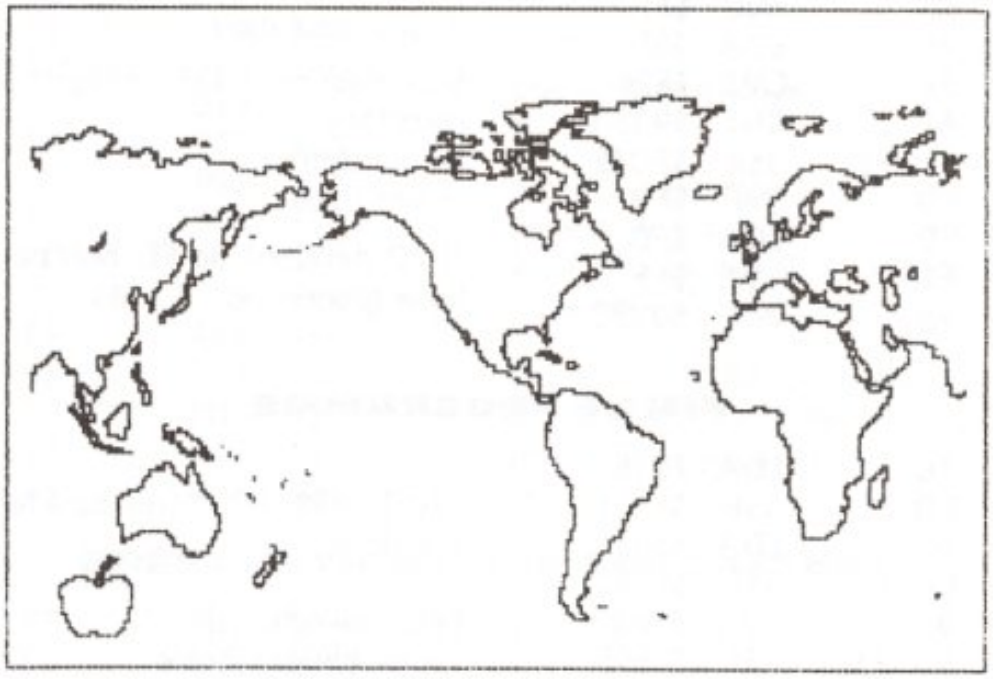
```
3AE :  0D 5A 09 5A 1B 36 31 54 1B 6E 1B 30 31 30 4C 1B 3E
3BF :  1B 0D 4E 30 38 09 0D 30 38 32 30 47 1B 00 00
```

BSAVE COPHGR.C,A$300,L206



Ci-dessus : **QUALITÉ**

Ci-dessous : **RAPIDE**



OCCUPATION

EXCLUSIVEMENT SOUS PRODOS

Votre disquette étant insérée dans le lecteur (port 6) de votre Apple, vous lancerez cette routine par BRUN OCCUPATION (ou CALL 768) si elle est déjà installée. Vous visualiserez alors la table d'occupation de la disquette. L'astérisque (*) indique un bloc occupé et le point (.) un bloc libre.

LA DISQUETTE Une disquette 5 pouces 1/4 possède 35 pistes (Ø à 34 ou $Ø à $22). Chaque piste comprend 8 blocs ce qui nous donne un total de 28Ø blocs (Ø à 279). Un bloc représente 512 octets. C'est le bloc 6 qui est réservé à la table d'occupation de la disquette (volume BIP.MAP).

PRÉSENTATION DE LA TABLE

```
300 :  20 58 FC    JSR  $FC58
303 :  A9 04       LDA  £$04
305 :  20 47 F8    JSR  $F847
308 :  A0 28       LDY  £$28
30A :  A2 33       LDX  £$33
30C :  CA          DEX
30D :  8A          TXA
30E :  88          DEY
30F :  91 26       STA  ($26),Y
311 :  C0 25       CPY  £$25
313 :  F0 F7       BEQ  $030C
315 :  C0 15       CPY  £$15
317 :  F0 F3       BEQ  $030C
319 :  C0 05       CPY  £$05
31B :  D0 F1       BNE  $030E
31D :  A0 84       LDY  £$84
31F :  8A          TXA
320 :  C8          INY
321 :  91 26       STA  ($26),Y
323 :  C9 39       CMP  £$39
325 :  D0 02       BNE  $0329
327 :  A2 00       LDX  £$00
329 :  C9 06       CMP  £$06
32B :  D0 02       BNE  $032F
32D :  A2 2F       LDX  £$2F
32F :  E8          INX
330 :  C0 A7       CPY  £$A7
332 :  D0 EB       BNE  $031F
334 :  A9 80       LDA  £$80
336 :  85 26       STA  $26
```

HOME classique.

GBASCALC place l'adresse de la ligne Ø4 en $26-27. Sur le GS, LDY est inutile, Y ayant cette valeur au retour de GBASCALC.

On va afficher la ligne ci-après, en mode inverse :

ØØØØØØØØØØØØØØØØ1111111111111111222

Y = $84 évite de recourir à GBASCALC pour la ligne suivante.

On affiche le complément de la ligne précédente :

Ø123456789ABCDEFØ123456789ABCDEFØ12

Ces deux lignes nous permettront de lire (verticalement) le numéro hexa de chaque piste (de $Ø à $22).

On ajuste l'adresse de la ligne (partie basse).

(suite page 32)

```
338 : A2 06      LDX  £$06
33A : E6 26      INC  $26
33C : 8A         TXA
33D : 20 C1 FB   JSR  $FBC1
340 : A0 04      LDY  £$04
342 : B1 26      LDA  ($26),Y
344 : 91 28      STA  ($28),Y
346 : E8         INX
347 : E0 0E      CPX  £$0E
349 : D0 EF      BNE  $033A
```

0 1 2 3 4 5 6 7

On désire cette fois afficher (toujours en mode inverse) les numéros des blocs. Il y en a 8 par piste (0 à 7).

OCCUPATION

```
34B : 20 00 BF   JSR  $BF00
34E : 80
34F : 7C 03
```

Appel classique au MLI.
$80 est le code de la fonction READ BLOCK MLI.
$37C est l'adresse des paramètres.

```
351 : A0 06      LDY  £$06
353 : 84 07      STY  $07
```

Numéro de la ligne d'affichage dans $7.

```
355 : A2 00      LDX  £$00
```

X à 0 pour lire le buffer.

```
357 : A0 05      LDY  £$05
359 : 84 06      STY  $06
```

Position horizontale du buffer dans $6.

```
35B : A5 07      LDA  $07
35D : 20 47 F8   JSR  $F847
```

GBASCALC place l'adresse de la ligne dans $26-27.

```
360 : A4 06      LDY  $06
```

Y est chargé avec la position horizontale.

```
362 : A9 AA      LDA  £$AA
```

* dans A pour bloc occupé.

```
364 : 1E 00 20   ASL  $2000,X
367 : 90 02      BCC  $036B
```

Si C = 0, bloc occupé. La valeur mise dans A (*) est bonne. Saut.

```
369 : A9 AE      LDA  £$AE
```

Sinon l'* est remplacée par un point ($AE).

```
36B : 91 26      STA  ($26),Y
```

Le symbole est poqué à la bonne adresse-écran.

```
36D : E8         INX
36E : C8         INY
```

X et Y incrémentés.

```
36F : C0 28      CPY  £$28
371 : 90 E6      BCC  $0359
```

Si Y est plus petit que $28, on continue.

```
373 : E6 07      INC  $07
```

Sinon il faut incrémenter pour la suite.

```
375 : A9 0E      LDA  £$0E
377 : C5 07      CMP  $07
```

Comparaison pour voir si la limite est atteinte.

```
379 : D0 DA      BNE  $0355
```

Non, on repart pour un nouveau tour !

```
37B : 60         RTS
```

Terminé !

PARAMÈTRES

```
37C : 03
37D : 60
37E : 00 20
380 : 06 00
```

Le MLI attend 3 paramètres :

① Le numéro du lecteur (1 octet) : S1,D1 = $60 - D2 = $E0
② L'adresse du buffer (2 octets). Ici : $2000.
③ Numéro du bloc à lire (2 octets). Ici : $6.

MISE EN GARDE Si vous remaniez ce programme, méfiez-vous de l'adresse $34E. Elle contient en effet le code de la fonction READ BLOCK MLI... sans danger, mais le code suivant ($81) concerne WRITE BLOCK et une erreur serait grave pour votre disquette...

OCCUPAMP

Le même utilitaire peut être utilisé avec l'Ampersand (&). On l'appellera alors par **&,S,D** où **S** = numéro du lecteur et **D** = numéro du Drive.

```
300 :  A9 0B      LDA  £$0B
302 :  8D F6 03   STA  $03F6
305 :  A9 03      LDA  £$03
307 :  8D F7 03   STA  $03F7
30A :  60         RTS
30B :  20 4C E7   JSR  $E74C
30E :  8A         TXA
30F :  0A         ASL
310 :  0A         ASL
311 :  0A         ASL
312 :  0A         ASL
313 :  48         PHA
314 :  20 F5 E6   JSR  $E6F5
317 :  68         PLA
318 :  E0 01      CPX  £$01
31A :  F0 02      BEQ  $031E
31C :  09 80      ORA  £$80
31E :  8D 9E 03   STA  $039E
```

Mise en place des vecteurs de l'Ampersand.

CHKCOM teste la virgule et évalue X.

Où $6 devient $60.

A empilé.

Appel à GETBYTC... pour changer !

A dépilé.

Si DRIVE 1, rien à faire...

Sinon bit 7 mis à 1 pour DRIVE 2.

Mise à jour de la table des paramètres.

La suite est identique au programme OCCUPATION, mais certaines adresses sont évidemment différentes, d'où ces nouveaux codes :

```
321 :  20 58 FC A9 04 20 47 F8 A0 28 A2 33 CA 8A 88
330 :  91 26 C0 25 F0 F7 C0 15 F0 F3 C0 05 D0 F1 A0 84
340 :  8A C8 91 26 C9 39 D0 02 A2 00 C9 06 D0 02 A2 2F
350 :  E8 C0 A7 D0 EB A9 80 85 26 A2 06 E6 26 8A 20 C1
360 :  FB A0 04 B1 26 91 28 E8 E0 0E D0 EF 20 00 BF 80
370 :  9D 03 A0 06 84 07 A2 00 A0 05 84 06 A5 07 20 47
380 :  F8 A4 06 A9 AA 1E 00 20 90 02 A9 AE 91 26 E8 C8
390 :  C0 28 90 E6 E6 07 A9 0E C5 07 D0 DA 60 03 60 00
3A0 :  20 06 00
```

En utilisant GETBYTC (qui saute un caractère et évalue X) au lieu de CHKCOM, on pourrait appeler la routine par **&(6(1** ou encore **&?6?1** ou tout simplement **&S6D1**, le plus pratique ! ■

SÉRIE-ALEA

Il est parfois intéressant de disposer d'une série aléatoire "sûre". Voici une solution qui présente certains avantages, notamment si la série ne dépasse pas 140 éléments, ce qui permet de la stocker page 3, à la suite de la routine.

Dans ce cas, on mémorisera la routine + la série par un **BSAVE SERIAL, A$300, L$CF**, ce qui permettra, lors de l'utilisation suivante, de repartir sur les bases ainsi mémorisées (après un BLOAD SERIAL, bien évidemment).

Un moyen plus "mathématique" pour obtenir un nombre aléatoire compris entre 0 et N. Au départ, N est dans $6. On trouve le résultat dans Y ou $7.

```
300 :  20 AE EF   JSR $EFAE
303 :  20 63 EB   JSR $EB63
306 :  A4 06      LDY $06
308 :  20 01 E3   JSR $E301
30B :  20 82 E9   JSR $E982
30E :  20 52 E7   JSR $E752
311 :  84 07      STY $07
313 :  60         RTS
```

$EFAE (RND) génère le nombre
$EB63 (MOVAF) place l'argument dans FAC
$E301 (SGNFLT) rend flottant l'entier sans signe dans Y
$E982 (FMULTT) multiplie FAC par ARG
$E752 (GETADR) transforme FAC en 2 octets (ici, un seul nous intéresse : celui de Y).

```
10 FOR I = 768 TO 833: READ R: POKE I,R: NEXT
15 DATA 32,245,230,134,6,138,208,14,32,245,230,134,6,157,66,3,202,138,16,249,166,6,202,169,255,197,6,144,3,74,128,249,133,7,218,32,174,239,165,161,37,7,197,6,176,245,250,168,185,66,3,72,189,66,3,153,66,3,104,157,66,3,202,16,225,96
20 TEXT : HOME : PRINT CHR$ (21):D$ = CHR$ (4)
25 CALL 768)0)21 : REM UNIQUEMENT POUR INITIALISER
30 D = 834: H = 1: HOME
35 FOR I = 0 TO 19: VTAB I + 1: HTAB H:V$ = "  ": IF PEEK (D + I) > 9 THEN V$ = ""
40 PRINT V$; PEEK (D + I): NEXT
45 CALL 768)21 : REM UTILISATION NORMALE
50 H = H + 4: IF H < 38 THEN 35
55 CALL - 198: POKE 49168,0: WAIT 49152, 128: POKE 49168,0
60 IF PEEK (49152) <> 27 THEN 30
65 VTAB 23: PRINT "(M)ENU DE DISQUETTE (A)PPLESOFT ";: GET R$: VTAB 22: PRINT
70 IF R$ = "M" THEN PRINT D$"RUN MENU"
75 IF R$ = "A" THEN END
80 GOTO 65
```

```
300 : 20 F5 E6   JSR  $E6F5   ]  GETBYTC saute un caractère. On récupère la
303 : 86 06      STX  $06     ]  valeur suivante dans X.
305 : 8A         TXA          ]  Si ce n'est pas 0, pas d'initialisation.
306 : D0 0E      BNE  $0316   ]
```

INITIALISATION

```
308 : 20 F5 E6   JSR  $E6F5   ]  On attend le nombre d'éléments + 1.
30B : 86 06      STX  $06     ]
30D : 9D 42 03   STA  $0342,X ]  Série de base, à partir de $342 : 0,1 etc.
310 : CA         DEX          ]  Ce "buffer" peut être installé à une autre
311 : 8A         TXA          ]  adresse et c'est indispensable s'il dépasse
312 : 10 F9      BPL  $030D   ]  142 éléments.
```

MÉLANGE

```
314 : A6 06      LDX  $06     Le nombre + 1 est dans 6.
316 : CA         DEX          X = X - 1.
317 : A9 FF      LDA  £$FF    ]
319 : C5 06      CMP  $06     ]
31B : 90 03      BCC  $0320   ]  Petit subterfuge pour déterminer la valeur
31D : 4A         LSR          ]  du AND utilisé plus loin.
31E : 80 F9      BRA  $0319   ]
320 : 85 07      STA  $07     ]
322 : DA         PHX          X est empilé (ne pas essayer sur le vieux 6502 !).
323 : 20 AE EF   JSR  $EFAE   RND.
326 : A5 A1      LDA  $A1     Le nombre (0 à 255) est récupéré dans $A1
                              (FAC LO).
328 : 25 07      AND  $07     Premier traitement qui le dégrossit.
32A : C5 06      CMP  $06     Comparaison avex max.
32C : B0 F5      BCS  $0323   S'il est égal ou plus grand, il est refusé.
32E : FA         PLX          X dépilé.
32F : A8         TAY          RND passé dans Y.
330 : B9 42 03   LDA  $0342,Y Lecture du nombre en $342 + Y.
333 : 48         PHA          Empilé.
334 : BD 42 03   LDA  $0342,X ]  Lecture du nombre en $342 + X...
337 : 99 42 03   STA  $0342,Y ]  et écriture immédiate en $342 + Y
33A : 68         PLA          A dépilé.
33B : 9D 42 03   STA  $0342,X Dernière phase de l'échange.
33E : CA         DEX          ]  X = X - 1
33F : 10 E1      BPL  $0322   ]  et la boucle continue jusqu'à 0 inclus.
341 : 60         RTS          FIN.
```

SYNTAXE : — INITIALISATION : CALL 768(0(N — MÉLANGE : CALL 768(N

N = Nombre d'éléments + 1.

Pour tirer parti de la série à partir du Basic, on s'inspirera de la ligne 35.

LIMITE : 142 en page 3 et 255 dans tous les cas. ■

TABBUF.DÉMO

Comment fonctionne la BUFFER TABLE de ProDOS ? Essayez cette petite démo. Elle vous éclairera sur les mystères de ProDOS. Attention ! Il faut un Buffer pour lire un CATALOG. Ne pas oublier que CLOSE fonctionne aussi en mode direct.

```
10 TEXT : NORMAL : PRINT CHR$ (21): HOME
15 FOR I = 768 TO 808: READ R: POKE I,R: NEXT : DATA 160,0,190,112,191,200,185,112,191,90,32,36,
   237,32,98,252,122,200,192,15,144,236,96,169,112,133,60,169,191,133,61,133,63,169,127,133,
   62,32,179,253,96
20 ONERR GOTO 70
25 D$ = CHR$ (4):I = 0
30 I = I + 1:F$ = "F" + STR$ (I): INVERSE : PRINT "OPEN  "F$: NORMAL : PRINT
35 PRINT D$"OPEN"F$: CALL 768: PRINT : CALL 791: PRINT : PRINT : PRINT : IF I <10 THEN 30
40 FOR I = 1 TO 2000: NEXT : HOME : PRINT D$"CAT"
45 CALL 791: PRINT : PRINT : PRINT
50 PRINT D$"CLOSE": FOR I = 1 TO 10:F$ = "F" + STR$ (I): PRINT D$"DELETE"F$: NEXT : PRINT
   D$"CAT"
55 CALL 791: PRINT : PRINT : CALL 768: PRINT : PRINT
60 VTAB 24: PRINT "(M)ENU DE DISQUETTE ";: GET R$: IF R$ = "M" THEN PRINT D$"RUN MENU"
65 HOME : END
70 PRINT "ERREUR: 8 FICHIERS DEJA OUVERTS:": PRINT
75 CALL 791: PRINT : PRINT : LIST 80
80 PRINT D$"CLOSE":I = I - 1: GOTO 30
```

```
300 :  A0 00      LDY  £$00
302 :  BE 70 BF   LDX  $BF70,Y
305 :  C8         INY
306 :  B9 70 BF   LDA  $BF70,Y
309 :  5A         PHY
30A :  20 24 ED   JSR  $ED24
30D :  20 62 FC   JSR  $FC62
310 :  7A         PLY
311 :  C8         INY
312 :  C0 0F      CPY  £$0F
314 :  90 EC      BCC  $0302
316 :  60         RTS
317 :  A9 70      LDA  £$70
319 :  85 3C      STA  $3C
31B :  A9 BF      LDA  £$BF
31D :  85 3D      STA  $3D
31F :  85 3F      STA  $3F
321 :  A9 7F      LDA  £$7F
323 :  85 3E      STA  $3E
325 :  20 B3 FD   JSR  $FDB3
328 :  60         RTS
```

Lecture et affichage de chaque adresse. La BUFFER TABLE va de BF70 à BF7F.

Retour chariot.
Y dépilé.

Boucle aussi longtemps que Y est plus petit que $F.

Retour.

Pour afficher les mémoires (de $BF70 à $BF7F), il faut placer l'adresse d'origine dans A1 ($3C-3D) et l'adresse de fin dans A2 ($3E-3F). $FDB3 se charge du reste.

Retour. ■

COPHGR.C1

Cette seconde version de COPIE.HGR autorise une reproduction en simple ou en double format.

Elle peut être appelée par CALL 24576,P1,P2,P3,P4,P5 ou &,P1,P2,P3,P4,P5 après initialisation de l'Ampersand par les deux pokes de la ligne 10.

Autre syntaxe : &)P1)P2)P3)P4)P5

P1 = 0 Photo - 1 Dessin
P2 = 0 Simple - 1 Double
P3 = 0 à 23 ligne de départ (VTAB - 1)
P4 = 1 à 24 ligne d'arrivée (VTAB)
P5 = 1 HGR - 2 HGR 2

P4 doit être supérieur à P3.

Il est possible d'améliorer l'impression en mettant l'imprimante en gras par un PRINT CHR$ (27)"!"
Mais ce n'est pas forcément d'un bel effet et il est préférable de travailler avec un ruban en bon état.

```
10 TEXT : HOME :D$ = CHR$ (4): PRINT
   CHR$ (21): PRINT D$"PREFIX/R4/
   COPIE.HGR": PRINT D$"BLOAD COP
   HGR.C1": POKE 1014,0: POKE 1015,
   96: REM &AMPERSAND = $6000

15 HOME : PRINT D$"CAT,TBIN"

20 PRINT : PRINT : PRINT : VTAB 22:
   INPUT "TITRE (OU RETURN) ";T$: IF
   T$ < > "" THEN PRINT D$ "BLOAD"
   T$",A$2000"

25 HOME : VTAB 22: PRINT "(0) PHOTO
   (RETURN) DESSIN ";: GET R$:I = 1:
   IF R$ = "0" THEN I = 0

30 HOME : VTAB 22: PRINT "(1) PETIT
   FORMAT (2) GRAND (0) FIN ";: GET
   R$: IF R$ > "2" THEN 30

35 ON VAL (R$) + 1 GOTO 50,40,45

40 & ,I,0,0,24,1: GOTO 15

45 & ,I,1,0,24,1: GOTO 15

50 HOME : VTAB 22: PRINT "(M)ENU DE
   DISQUETTE ";: GET R$

55 IF R$ = "M" THEN PRINT D$"RUN
   /R4/MENU"

60 HOME
```

MARGE : Elle est fixée à 1 pour permettre l'impression en double largeur. Pour la modifier, s'inspirer de cette ligne :

```
M$ = "L030" : X=0: FOR I = 24833 TO 24830 STEP - 1: X =
X + 1 : POKE I, ASC (MID$(M$,X,1)): NEXT
```

(suite page 38)

PARAMÈTRES

```
6000 : A0 00      LDY £$00
6002 : 5A         PHY
6003 : 20 F5 E6   JSR $E6F5
6006 : 7A         PLY
6007 : 96 06      STX $06,Y
6009 : C8         INY
600A : C0 05      CPY £$05
600C : 90 F4      BCC $6002
```

On a successivement :
0 (PHOTO) ou 1 (DESSIN)
0 (SIMPLE FORMAT) ou 1 (DOUBLE)
Fourchette pour la partie de l'image à imprimer (0,24)
1 (PAGE 1) ou 2 (PAGE 2).

PAGE 1 ou PAGE 2

```
600E : A9 1C      LDA £$1C
6010 : 66 0A      ROR $0A
6012 : B0 02      BCS $6016
6014 : A9 3C      LDA £$3C
6016 : 85 0A      STA $0A
```

$1C pour HGR
$3C pour HGR2
dans A.

IMPRIMANTE INITIALISÉE

```
6018 : A9 01      LDA £$01
601A : 20 95 FE   JSR $FE95
601D : A2 17      LDX £$17
601F : BD F3 60   LDA $60F3,X
6022 : 20 ED FD   JSR $FDED
6025 : CA         DEX
6026 : 10 F7      BPL $601F
```

CHR$ (9)"80N"
CHR$ (27)" > " (unidirectionnel)
CHR$ (27)"001" (marge)
CHR$ (27)"n" (caractère)
CHR$ (27)"T16" (interligne)
CHR$ (27)"Z" (pas d'avancement de ligne
CHR$ (9)"Z" en cas de dépassement).

G0280 ou G0560

```
6028 : A2 07      LDX £$07
602A : A4 07      LDY $07
602C : F0 05      BEQ $6033
602E : BD 13 61   LDA $6113,X
6031 : 80 03      BRA $6036
6033 : BD 0B 61   LDA $610B,X
6036 : 20 ED FD   JSR $FDED
6039 : CA         DEX
603A : 10 EE      BPL $602A
```

Format simple : G0280
Format double : G0560

LONGUEUR DE LIGNE

```
603C : E8         INX
603D : 86 FF      STX $FF
```

X = FF. On le met à 0 pour longueur de ligne ($FF).

ADRESSE DE LA LIGNE

```
603F : A5 08      LDA $08
6041 : 20 47 F8   JSR $F847
6044 : 65 FF      ADC $FF
6046 : 85 0B      STA $0B
6048 : A5 27      LDA $27
604A : 65 0A      ADC $0A
604C : AC 58 60   LDY $6058
604F : C0 04      CPY £$04
```

GBASCALC place l'adresse de la ligne dans $26/27, mais la partie basse est aussi dans A. On y ajoute immédiatement la position dans la ligne (0 au départ). Résultat dans $B.
Partie haute de l'adresse plus la page ($1C ou $3C).
Si on lit 4 à cette adresse, on est en double format et on traite la...

```
6051 : D0 02     BNE $6055   ┐ deuxième partie de la ligne. Sinon, saut.
6053 : 69 0B     ADC £$0B    ┐ Addition (normalement 3 fois 4... mais il y a
6055 : 85 0C     STA $0C     ┘ une retenue). La partie haute est en place.
```

ON TRAITE UN CARACTÈRE

```
6057 : A0 08     LDY £$08      8 lignes au départ.
6059 : A2 00     LDX £$00      X à 0 pour commencer.
605B : A1 0B     LDA ($0B,X)   Lecture de l'octet $R/C+X.
605D : 4A        LSR           Décalage à droite.
605E : 76 18     ROR $18,X     Rotation à droite (la retenue poussée par
                               LSR rentre à gauche).
6060 : 48        PHA           A empilé.
6061 : A5 07     LDA $07     ┐ Si on est en format simple, saut.
6063 : F0 07     BEQ $606C   ┘
6065 : B5 18     LDA $18,X   ┐
6067 : 16 18     ASL $18,X   │ Sinon, petite gymnastique amusante
6069 : 6A        ROR         │ pour s'offrir deux fois la même retenue
606A : 95 18     STA $18,X   ┘ (ligne répétée pour double hauteur).
606C : 68        PLA           A est récupéré.
606D : E8        INX         ┐
606E : E0 07     CPX £$07    │ Si X est inférieur à 7, on continue les
6070 : 90 EB     BCC $605D   ┘ décalages.
```

```
6072 : A9 03     LDA £$03    ┐ +3 pour la ligne suivante (il faut norma-
6074 : 65 0C     ADC $0C     │ lement 4 comme $400=1024, mais la
6076 : 85 0C     STA $0C     ┘ retenue est à 1 en sortant de la boucle).
6078 : A5 07     LDA $07     ┐ Si simple format, saut.
607A : F0 09     BEQ $6085   ┘
607C : C0 05     CPY £$05    ┐ Sinon, Y est-il à 5 ? Oui : on en a fini
607E : F0 08     BEQ $6088   ┘ avec le premier groupe de 4 lignes... que
                               nous avons répétées pour faire 8.
6080 : 88        DEY         ┐ Traitement pour format double.
6081 : 10 D6     BPL $6059   ┘
6083 : 30 03     BMI $6088
6085 : 88        DEY         ┐ Traitement pour format simple.
6086 : D0 D1     BNE $6059   ┘
```

ON IMPRIME

```
6088 : A2 FF     LDX £$FF      X = $FF.
608A : E8        INX           Puis 0 au départ.
608B : B5 18     LDA $18,X     Lecture du tampon.
608D : A4 06     LDY $06     ┐
608F : 88        DEY         │ Si mode 0, EOR 11111111 inverse le
6090 : F0 02     BEQ $6094   │ caractère.
6092 : 49 FF     EOR £$FF    ┘
6094 : 48        PHA           Valeur A empilée.
6095 : 09 80     ORA £$80    ┐ Traitement éventuel des caractères
6097 : C9 8D     CMP £$8D    │ "vicieux" : $D et $8D, $1A et $9A (au
6099 : F0 04     BEQ $609F   │ préalable, ORA 1000 0000 force le bit
609B : C9 9A     CMP £$9A    ┘ 7 à 1).
```

(suite page 40)

```
609D : D0 04      BNE $60A3
609F : 68         PLA
60A0 : 29 F7      AND £$F7
60A2 : 48         PHA
60A3 : 68         PLA
```

Valeur réelle de A récupérée.
AND 11110111 force le bit 3 à 0 et élimine le "vice".
Pour le bon équilibre de la pile.
Récupération définitive (caractère modifié ou non).

```
60A4 : 20 ED FD   JSR $FDED
60A7 : A4 07      LDY $07
60A9 : F0 03      BEQ $60AE
60AB : 20 ED FD   JSR $FDED
60AE : E0 06      CPX £$06
60B0 : D0 D8      BNE $608A
```

IMPRESSION.
Si format simple, saut.
Sinon, nouvelle impression.
Si X est différent de 6, on continue.

```
60B2 : E6 FF      INC $FF
60B4 : A5 FF      LDA $FF
60B6 : C9 28      CMP £$28
60B8 : 90 85      BCC $603F
```

Pointeur de la position dans la ligne, incrémenté. On continue jusqu'à $27 inclus (0 à $27... ou 40 × 7 = 280).

NOUVELLE LIGNE ?

```
60BA : 20 8E FD   JSR $FD8E
60BD : A5 07      LDA $07
60BF : F0 0F      BEQ $60D0
60C1 : AC 58 60   LDY $6058
60C4 : C0 08      CPY £$08
60C6 : D0 05      BNE $60CD
60C8 : 4E 58 60   LSR $6058
60CB : 80 0B      BRA $60D8
60CD : 0E 58 60   ASL $6058
60D0 : E6 08      INC $08
60D2 : A5 08      LDA $08
60D4 : C5 09      CMP $09
60D6 : B0 03      BCS $60DB
60D8 : 4C 28 60   JMP $6028
```

RETURN.
Si simple impression, saut.
Si l'octet témoin est différent de 8, on saute à ASL pour le multiplier par 2 et en faire 8 (car il est à 4).
Si c'est 8, LSR le divise par 2 et en fait 4... et en route pour la deuxième moitié de la ligne.
Décalage à gauche que multiplie 4 par 2.
Si toutes les lignes ont été traitées, c'est terminé.
Sinon, on continue...

FIN

```
60DB : A9 1B      LDA $1B
60DD : 20 ED FD   JSR $FDED
60E0 : A9 63      LDA £$63
60E2 : 20 ED FD   JSR $FDED
60E5 : A9 03      LDA £$03
60E7 : 20 95 FE   JSR $FE95
60EA : A9 F1      LDA £$F1
60EC : 85 36      STA $36
60EE : A9 B7      LDA £$B7
60F0 : 85 37      STA $37
60F2 : 60         RTS
```

PRINT CHR$ (27)"c" restaure les instructions standard.
PR£3... mais ça ne suffit pas.
ProDOS retombera ainsi sur ses pattes.
BASIC.

CODES IMPRIMANTE

```
60F3 : 0D 5A 09 5A 1B 36 31 54 1B 6E 1B 31 30 30 4C 1B 3E 1B 0D 4E 30
6108 : 38 09 0D 30 38 32 30 47 1B 00 00 30 36 35 30 47 1B 00 00
```

QUOI40.80

```
10 FOR I = 768 TO 817: READ R: POKE I,R: NEXT
15 DATA 32,76,231,218,32,76,231,202,138,32,71,248,250,138,44,31,192,16,5,
   201,2,144,1,74,101,38,133,38,198,38,44,31,192,16,7,138,74,144,3,44,85,
   192,177,38,44,84,192,133,6,96
20 TEXT : NORMAL :D$ = CHR$ (4): PRINT D$"PR£3": PRINT : HOME
25 HOME : PRINT "Q U O I  E S T  O U ?"
30 PRINT " ____________________": PRINT
35 PRINT "HTAB3:VTAB1, IL Y A: ";: CALL 768,3,1: PRINT CHR$ ( PEEK
   (6))
40 PRINT : PRINT "HTAB5:VTAB4, IL Y A: ";: CALL 768,5,4: PRINT CHR$
   ( PEEK (6))
45 VTAB 7: PRINT : CALL - 958: PRINT "(8)0 COLONNES (4)0 COLON
   NES ";: GET R$: IF R$ = "8" THEN PRINT CHR$ (18): GOTO 25
50 IF R$ = "4" THEN PRINT CHR$ (17): GOTO 25
55 FOR I = 1 TO 12
60 VTAB 7: PRINT : HTAB 1: CALL - 958: INPUT "H, V ";H,V
65 IF NOT H OR NOT V OR (V = 24 AND H = 80) OR (V = 24 AND H =
   40) OR V > 24 THEN 60
70 VTAB 9: CALL 768,H,V: PRINT PEEK (6): POKE 1403,H - 1: VTAB V:
   INVERSE : PRINT " ";: NORMAL : GOSUB 90: NEXT
75 VTAB 22: PRINT : PRINT "(M)ENU DISQUETTE (C)HANGER ";: GET R$:
   IF R$ = "C" THEN 45
80 IF R$ = "M" THEN PRINT D$"RUN MENU"
85 END
90 CALL - 198: POKE 49168,0: WAIT 49152,128: POKE 49168,0: IF PEEK
   (49152) = 27 THEN POP : GOTO 75
95 RETURN
```

```
00/0300 :  20 4C E7   JSR  E74C   ┐ COMBYTE évalue l'expression. On
00/0303 :  DA         PHX         ┘ a le résultat dans X qui est empilé.
00/0304 :  20 4C E7   JSR  E74C   ┐ Deuxième paramètre.
00/0307 :  CA         DEX         ┘ X = X - 1 pour ajuster CV.
00/0308 :  8A         TXA         ┐ GBASCALC place l'adresse de la
00/0309 :  20 47 F8   JSR  F847   ┘ ligne en $26-27.
00/030C :  FA         PLX         ┐ Récupération de la position CH sur la
00/030D :  8A         TXA         ┘ pile. On la passe dans A.
```

(suite page 42)

```
00/030E :  2C 1F C0     BIT  C01F   Si on n'est pas en 80 colonnes,
00/0311 :  10 05        BPL  0318   saut.
00/0313 :  C9 02        CMP  #02    Si CH plus petit que 2, saut itou.
00/0315 :  90 01        BCC  0318
00/0317 :  4A           LSR         Sinon décalage à droite qui divise A par
00/0318 :  65 26        ADC  26     deux. S'il y a un reste, il est dans la retenue.
00/031A :  85 26        STA  26     On ajuste la position horizontale.
00/031C :  C6 26        DEC  26
00/031E :  2C 1F C0     BIT  C01F   Si on n'est pas en 80 colonnes,
00/0321 :  10 07        BPL  032A   saut.
00/0323 :  8A           TXA         X dans A.
00/0324 :  4A           LSR         Décalage à droite.
00/0325 :  90 03        BCC  032A   S'il n'y a pas de retenue, saut.
00/0327 :  2C 55 C0     BIT  C055   Sinon il faut lire la colonne paire.
00/032A :  B1 26        LDA  (26),Y Lecture.
00/032C :  2C 54 C0     BIT  C054   Retour éventuel en MEM NORMALE.
00/032F :  85 06        STA  06     Stockage en 6.
00/0331 :  60           RTS         Basic.
```

PRT.USING

PRINT USING élémentaire

```
10  FOR I = 768 TO 808: READ R: POKE I,R: NEXT
15  DATA  32,245,230,134,9,32,190,222,32,227,223,160,2,177,131,153,6,0,136,16,248,200,196,6,240,6,
    177,7,201,46,208,245,165,9,132,9,229,9,133,36,96
20  TEXT : NORMAL : PRINT CHR$ (21): HOME
25  FOR I = 1 TO 4: READ R:A$ = STR$ (R): CALL 768,9,A$: PRINT A$;: READ R:A$ = STR$ (R):
      CALL 768,24,A$: PRINT A$: NEXT
30  VTAB 22: PRINT "(M)ENU DE DISQUETTE ";: GET R$: IF R$ = "M" OR R$ = "m" THEN PRINT
    CHR$ (4)"RUN MENU"
35  HOME : END
40  DATA 0.23,123,1230.1,235.10,78649.80,.17,1,75.2
```

```
300 :  20 F5 E6     JSR  $E6F5   HTAB en $9.
303 :  86 09        STX  $09
305 :  20 BE DE     JSR  $DEBE   Le nombre a préalablement été trans-
308 :  20 E3 DF     JSR  $DFE3   formé en variable alphanumérique. Son
30B :  A0 02        LDY  #$02    adresse est en $7.8 et sa longueur en $6.
30D :  B1 83        LDA  ($83),Y
30F :  99 06 00     STA  $0006,Y
312 :  88           DEY
313 :  10 F8        BPL  $030D
315 :  C8           INY          Y = 0.
316 :  C4 06        CPY  $06     Si Y = longueur, terminé.
318 :  F0 06        BEQ  $0320
31A :  B1 07        LDA  ($07),Y Est-ce le point ?
31C :  C9 2E        CMP  #$2E
31E :  D0 F5        BNE  $0315   Non : on continue.
320 :  A5 09        LDA  $09     Oui, on détermine HTAB réel.
322 :  84 09        STY  $09     (Valable en 40 colonnes).
324 :  E5 09        SBC  $09
326 :  85 24        STA  $24
328 :  60           RTS          Basic.
```

ÉCHANGE

HGR devient HGR2
et vice versa

```
100 TEXT : NORMAL :D$ = CHR$ (4): PRINT CHR$ (21): HOME
105 FOR I = 768 TO 802: READ R: POKE I,R: NEXT
110 DATA  160,0,132,6,132,8,169,32,170,133,7,10,133,9,177,6,72,177,8,145,6,
    104,145,8,200,208,243,230,7,230,9,202,208,236,96
115 GOSUB 160: GOSUB 160
120 X = 1: GOSUB 195
125 POKE 49232,0: POKE 49234,0: POKE 49236,0: POKE 49239,0
130 GOSUB 200: TEXT :X = 2: GOSUB 195
135 POKE 49232,0: POKE 49234,0: POKE 49237,0: POKE 49239,0
140 GOSUB 200: TEXT: VTAB 12: PRINT "ECHANGE DES PAGES
    (ESC=FIN)": GOSUB 200
145 IF PEEK (49152) < > 27 THEN CALL 768: GOTO 125
150 TEXT : HOME : VTAB 22: PRINT "(M)ENU DE DISQUETTE ";: GET R$: IF
    R$ = "M" THEN PRINT D$"RUN MENU"
155 HOME : END
160 HOME : PRINT D$"CAT,TBIN"
165 PRINT : PRINT : VTAB 23: INPUT "TITRE ? ":T$
170 VTAB 23: CALL - 958: PRINT "(1)HGR (2)HGR2 ";: GET R$: IF R$ =
    CHR$ (13) THEN R$ = "1"
175 IF NOT LEN (T$) THEN 190
180 A$ = ",A$" + STR$ ( VAL (R$) * 2) + "000"
185 PRINT D$"BLOAD"T$:A$
190 RETURN
195 HOME : VTAB 12: PRINT "UNE TOUCHE POUR PAGE "X
200 CALL - 198: POKE 49168,0: WAIT 49152,128: POKE 49168,0: RETURN
```

```
00/0300 :   A0 00     LDY £00
00/0302 :   84 06     STY 06
00/0304 :   84 08     STY 08
00/0306 :   A9 20     LDA £20
00/0308 :   AA        TAX
00/0309 :   85 07     STA 07
00/030B :   0A        ASL
00/030C :   85 09     STA 09
```

On installe l'adresse de la page HGR en $6/7 et celle de HGR2 en $8/9.
Noter que ASL multiplie A par deux et transforme donc $20 en $40 par un décalage à gauche.
A la sortie, X contient $20 et Y $0.

(suite page 44)

```
00/030E :  B1 06      LDA (06),Y   ┐ Lecture de l'octet HGR et sauve-
00/0310 :  48         PHA          │ garde sur la pile.
00/0311 :  B1 08      LDA (08),Y   │ Lecture de l'octet HGR2 et écriture
00/0313 :  91 06      STA (06),Y   │ immédiate en HGR.
00/0315 :  68         PLA          │ Récupération de A sur la pile et
00/0316 :  91 08      STA (08),Y   │ écriture en HGR2.
00/0318 :  C8         INY          │ On continue les permutations
00/0319 :  D0 F3      BNE 030E     ┘ jusqu'à Y = $FF inclus.
00/031B :  E6 07      INC 07       ┐ Plus 1 pour la partie haute des
00/031D :  E6 09      INC 09       ┘ deux adresses.
00/031F :  CA         DEX          ┐ On continuera jusqu'à X = 1 inclus.
00/0320 :  D0 EC      BNE 030E     ┘
00/0322 :  60         RTS            Retour au Basic.
```

Sauvegarde éventuelle : **BSAVE ECHANGE.C,A$300,L$23**

PRT.USVIRG

Même routine que PRT.USING, mais le point est transformé en virgule. La démo en Basic est identique, à l'exception, bien sûr, des lignes 10 et 15... que voici :

```
10  FOR I = 768 TO 812: READ R: POKE I,R: NEXT
15  DATA  32,245,230,134,9,32,190,222,32,227,223,160,2,177,131,153,6,0,136,16,248,
    200,196,6,240,10,177,7,201,46,208,245,169,44,145,7,165,9,132,9,229,9,133,36,96
```

```
300 :  20 F5 E6    JSR $E6F5     ┐ HTAB dans $9.
303 :  86 09       STX $09       ┘
305 :  20 BE DE    JSR $DEBE     ┐
308 :  20 E3 DF    JSR $DFE3     │
30B :  A0 02       LDY £$02      │ Le nombre (longueur et adresse)
30D:   B1 83       LDA ($83),Y   │ en $6-7-8.
30F :  99 06 00    STA $0006,Y   │
312 :  88          DEY           │
313 :  10 F8       BPL $030D     ┘
315 :  C8          INY             Y = 0.
316 :  C4 06       CPY $06         Si Y = longueur, terminé !
318 :  F0 0A       BEQ $0324     ┐
31A:   B1 07       LDA ($07),Y   │
31C :  C9 2E       CMP £$2E      │ On cherche le point éventuel.
31E :  D0 F5       BNE $0315     ┘
320 :  A9 2C       LDA £$2C      ┐ Si on le trouve, on le remplace par une
322 :  91 07       STA ($07),Y   ┘ virgule.
324 :  A5 09       LDA $09       ┐
326 :  84 09       STY $09       │ HTAB ajusté.
328 :  E5 09       SBC $09       │
32A:   85 24       STA $24       ┘
32C :  60          RTS             Basic.
```

■

HGR.UP

Vous l'avez compris : il s'agit tout simplement d'un défilement de la page HGR vers le haut. La première ligne remplace la dernière tandis que les 191 autres remontent d'un cran.
Cette version utilise une partie d'une routine Applesoft : HPOSN. Normalement, HPOSN positionne le curseur sans tracer de point, mais ici, nous avons seulement extrait la routine de calcul de l'adresse de chaque ligne graphique.

HGR.UPBAS

```
100 TEXT : NORMAL :D$ = CHR$ (4): PRINT CHR$ (21): HOME
105 HGR : TEXT : PRINT D$"BLOAD/R4/HGR/HGR.UP"
110 VTAB 10: PRINT "Presser une touche à chaque bip"
115 HCOLOR = 3: POKE 230,32: HPLOT 0,0 TO 279,0 TO 279,191 TO
    0,191 TO 0,0 TO 279,191: HPLOT 279,0 TO 0,191
120 GOSUB 170
125 POKE 49232,0: POKE 49234,0: POKE 49236,0: POKE 49239,0
130 GOSUB 170
135 CALL 768
140 GOSUB 170: TEXT
145 HOME : VTAB 22: PRINT "(E)NCORE (M)ENU DISQUETTE (F)IN ";:
    GET R$
150 IF R$ = "M" THEN PRINT D$"RUN /R4/MENU"
155 IF R$ = "E" THEN 120
160 IF R$ = "F" THEN HOME : END
165 GOTO 145
170 CALL - 198: POKE 49168,0: WAIT 49152,128: POKE 49168,0:
    RETURN
```

```
300 :  A2 00        LDX  £$C0
302 :  A9 20        LDA  £$20
304 :  85 E6        STA  $E6
306 :  85 07        STA  $07
308 :  A0 28        LDY  £$28
30A:   B9 FF 1F     LDA  $1FFF,Y
30D:   99 D7 02     STA  $02D7,Y
310 :  88           DEY
311 :  D0 F7        BNE  $030A
```

Pointeur du nombre de lignes.

$20 dans HPAG (pour HPOSN).

Et aussi dans $7.

Mémorisation (dans une petite partie du buffer) de la ligne 0.
A la sortie, Y = 0.

(suite page 46)

```
313 :  84 Ø6      STY  $Ø6        Dans $6 pour avoir $2ØØØ dans
                                  $6/7 (voir plus haut).
315 :  C8         INY
316 :  84 Ø8      STY  $Ø8        Pointeur nombre de lignes.
318 :  98         TYA             Y dans A pour CALCUL ADR.
```

HPOSN

```
319 :  48         PHA
31A:   29 CØ      AND  £$CØ
31C :  85 26      STA  $26
31E :  4A         LSR
31F :  4A         LSR
32Ø :  Ø5 26      ORA  $26
322 :  85 26      STA  $26
324 :  68         PLA             Admirez le travail.
325 :  85 27      STA  $27        A partir du seul numéro de ligne,
327 :  ØA         ASL             les auteurs de la ROM vous four-
328 :  ØA         ASL             nissent son adresse... sans sous-
329 :  ØA         ASL             traction, sans addition... et sans
32A:   26 27      ROL  $27        modifier le contenu des registres
32C :  ØA         ASL             X et Y.
32D :  26 27      ROL  $27
32F :  ØA         ASL
33Ø :  66 26      ROR  $26
332 :  A5 27      LDA  $27
334 :  29 1F      AND  £$1F
336 :  Ø5 E6      ORA  $E6
338 :  85 27      STA  $27
```

ÉCHANGE

```
33A:   AØ 27      LDY  £$27       $27 positions.
33C :  B1 26      LDA  ($26),Y    Le contenu de la ligne B passe
33E :  91 Ø6      STA  ($Ø6),Y    dans la A.
34Ø :  88         DEY
341 :  1Ø F9      BPL  $Ø33C      Jusqu'à Y = Ø inclus.
343 :  A5 26      LDA  $26
345 :  85 Ø6      STA  $Ø6        L'adresse de la ligne B devient
347 :  A5 27      LDA  $27        celle de la ligne A.
349 :  85 Ø7      STA  $Ø7
34B :  E6 Ø8      INC  $Ø8
34D :  A5 Ø8      LDA  $Ø8        Où en est-on ?
34F :  C9 CØ      CMP  £$CØ       Il y a 192 lignes (Ø à 191).
351 :  DØ C6      BNE  $Ø319      Pas terminé ?
```

DERNIÈRE LIGNE

```
353 :  AØ 27      LDY  £$27
355 :  B9 D8 Ø2   LDA  $Ø2D8,Y    Récupération de la ligne Ø stockée
358 :  99 DØ 3F   STA  $3FDØ,Y    dans le buffer et écriture en bas de
35B :  88         DEY             page.
35C :  1Ø F7      BPL  $Ø355
35E :  CA         DEX
35F :  DØ A1      BNE  $3Ø2       On continue le topo jusqu'à la
361 :  2C ØØ CØ   BIT  $CØØØ     limite prévue ($CØ).
364 :  1Ø 9A      BPL  $Ø3ØØ
366 :  6Ø         RTS             BASIC.
```

HGR.DOWN

Même routine que HGR.UP, mais avec défilement de l'écran vers le bas. On notera que quelques pokes permettent en fait de transformer le mouvement descendant en mouvement ascendant.

HGR.DOWNBAS

```
100 TEXT : NORMAL :D$ = CHR$ (4): PRINT CHR$ (21): HOME
105 HGR : TEXT : PRINT D$"BLOAD/R4/HGR/HGR.DOWN"
110 VTAB 10: PRINT "Presser une touche à chaque bip"
115 HCOLOR = 3: POKE 230,32: HPLOT 0,0 TO 279,0 TO 279,191 TO
    0,191 TO 0,0 TO 279,191: HPLOT 279,0 TO 0,191
120 GOSUB 175
125 POKE 49232,0: POKE 49234,0: POKE 49236,0: POKE 49239,0
130 GOSUB 175
135 CALL 768
140 GOSUB 175: TEXT
145 HOME : VTAB 22: PRINT "(E)NCORE (M)ENU DISQUETTE (U)P
    (F)IN ";: GET R$
150 IF R$ = "M" THEN PRINT D$"RUN /R4/MENU"
155 IF R$ = "E" THEN 200
160 IF R$ = "U" THEN 185
165 IF R$ = "F" THEN HOME : END
170 GOTO 145
175 CALL - 198: POKE 49168,0: WAIT 49152,128: POKE 49168,0:
    RETURN
180 REM Ces pokes transforment HGR.DOWN en HGR.UP
185 POKE 774,234: POKE 775,234: POKE 779,0: POKE 785,255: POKE
    786,31: POKE 794,0: POKE 847,230: POKE 852,192: POKE
    861,208: POKE 862,63
190 GOTO 125
195 REM Ces pokes rétablissent HGR.DOWN
200 POKE 774,169: POKE 775,63: POKE 779,208: POKE 785,207:
    POKE 786,63: POKE 794,190: POKE 847,198: POKE 852,255:
    POKE 861,0: POKE 862,32
205 GOTO 125
```

(suite page 48)

INITIALISATION

```
300 :  A2 C0      LDX  £$C0
302 :  A9 20      LDA  £$20
304 :  85 E6      STA  $E6
306 :  A9 3F      LDA  £$3F
308 :  85 07      STA  $07
30A :  A9 D0      LDA  £$D0
30C :  85 06      STA  $06
```

Pointeur nombre de lignes.

$20 dans HPAG pour HPOSN.

$3FD0 dans $6-7 (adresse de début de la dernière ligne de l'écran).

MÉMO DERNIÈRE LIGNE

```
30E :  A0 28      LDY  £$28
310 :  B9 CF 3F   LDA  $3FCF,Y
313 :  99 D7 02   STA  $02D7,Y
316 :  88         DEY
317 :  D0 F7      BNE  $0310
```

Mémorisation (dans la partie haute du buffer) de la ligne n°191 (dernière de l'écran graphique).

POINTEUR NUMÉRO DE LIGNE

```
319 :  A9 BE      LDA  £$BE
31B :  85 08      STA  $08
```

La première traitée portera le numéro 190.

ADRESSE DE LIGNE (HPOSN partiel)

```
31D :  48         PHA
31E :  29 C0      AND  £$C0
320 :  85 26      STA  $26
322 :  4A         LSR
323 :  4A         LSR
324 :  05 26      ORA  $26
326 :  85 26      STA  $26
328 :  68         PLA
329 :  85 27      STA  $27
32B :  0A         ASL
32C :  0A         ASL
32D :  0A         ASL
32E :  26 27      ROL  $27
330 :  0A         ASL
331 :  26 27      ROL  $27
333 :  0A         ASL
334 :  66 26      ROR  $26
336 :  A5 27      LDA  $27
338 :  29 1F      AND  £$1F
33A :  05 E6      ORA  $E6
33C :  85 27      STA  $27
```

Admirons le travail : déterminer l'adresse d'une ligne à partir de son numéro, sans aucune addition, soustraction ou appel à une autre routine de l'Apple.
C'est ce que fait cet extrait de HPOSN.
Notons que les registres X et Y ne sont pas davantage modifiés.

ÉCHANGE COMPLET

```
33E :  A0 27      LDY  £$27
340 :  B1 26      LDA  ($26),Y
342 :  91 06      STA  ($06),Y
344 :  88         DEY
345 :  10 F9      BPL  $0340
347 :  A5 26      LDA  $26
349 :  85 06      STA  $06
34B :  A5 27      LDA  $27
34D :  85 07      STA  $07
```

$28 (40) positions (0 à $27).
Lecture d'une ligne et écriture sur la ligne suivante.

Boucle.

L'adresse de la seconde devient celle de la première.

```
34F :  C6 08        DEC  $08      ┐ Nombre de lignes à traiter
351 :  A5 08        LDA  $08      ┘ décrémenté.
353 :  C9 FF        CMP  £$FF     ┐
355 :  D0 C6        BNE  $031D    ┘ On va boucler jusqu'à 0 inclus.
```

RÉCUPÉRATION DERNIÈRE LIGNE

```
357 :  A0 27        LDY  £$27     ┐
359 :  B9 D8 02     LDA  $02D8,Y  │ On récupère la dernière ligne dans le
35C :  99 00 20     STA  $2000,Y  │ buffer et on l'écrit à la place de la
35F :  88           DEY           │ première.
360 :  10 F7        BPL  $0359    ┘
362 :  CA           DEX           ┐ Quand X sera égal à 0, on aura
363 :  D0 9D        BNE  $0302    ┘ retrouvé l'écran initial.
```

ON CONTINUE ?

```
365 :  2C 00 C0     BIT  $C000    ┐
368 :  10 96        BPL  $0300    ┘ Suite ou arrêt*.
36A :  60           RTS
```

** On pourrait supprimer le compteur X ($300.301) et se contenter de ce test. Il suffirait alors de presser une touche pour arrêter n'importe où. Dans ce cas, il faudrait également supprimer DEX et BNE ($362/364). Si l'on ne désire pas incorporer ce test, placer au contraire un RTS ($60) en $365.*

VIRGULE

Remplacez le point des nombres décimaux par une virgule !

```
10 FOR I = 768 TO 796: READ R: POKE I,R: NEXT
15 DATA 32,190,222,32,103,221,32,52,237,189,0,1,240,12,201,46,208,5,169,44,157,0,1,
   232,208,239,76,58,219
20 TEXT : NORMAL : PRINT CHR$ (21): HOME
25 A = 9999
30 FOR I = 1 TO 8:A = A *  0.3: CALL 768,A: PRINT : NEXT
35 VTAB 22: PRINT "(M)ENU DE DISQUETTE ";: GET R$: IF R$ = "M" THEN PRINT CHR$
   (4)"RUN MENU"
40 HOME
```

```
300 :  20 BE DE     JSR  $DEBE    CHKCOM teste la virgule du CALL 768,A.
303 :  20 67 DD     JSR  $DD67    FRMNUM évalue la formule (valeur dans
                                  FAC).
306 :  20 34 ED     JSR  $ED34    FOUT convertit la valeur de FAC en chaîne.
309 :  BD 00 01     LDA  $0100,X  ┐
30C :  F0 0C        BEQ  $031A    │
30E :  C9 2E        CMP  £$2E     │ A la sortie de FOUT, X=0 et Y=1
310 :  D0 05        BNE  $0317    │ (A=0). On remplace le point éventuel
312 :  A9 2C        LDA  £$2C     │ par une virgule.
314 :  9D 00 01     STA  $0100,X  │ A la sortie de cette boucle, A=0.
317 :  E8           INX           │
318 :  D0 EF        BNE  $0309    ┘
31A :  4C 3A DB     JMP  $DB3A    STROUT affiche la chaîne pointée par Y - A (0 -
                                  1, c'est-à-dire $100) et se terminant par un 0.
```

HGR.RTN

L'IMAGE DE L'ÉCRAN HGR est retournée

HGR.RBAS

```
100  TEXT : NORMAL :D$ = CHR$ (4): PRINT CHR$ (21): HOME
101  PRINT D$"BLOAD/R4/COPIE.HGR/COPEX"
105  PRINT D$"BLOAD/R4/HGR/HGR.RTN"
110  VTAB 10: PRINT "Escape pour terminer"
120  GOSUB 175
125  POKE 49232,0: POKE 49234,0: POKE 49236,0: POKE 49239,0
130  FOR I = 1 TO 8: CALL 768: GOSUB 175: IF PEEK (49152) <> 27
       THEN NEXT
140  TEXT
145  HOME : VTAB 22: PRINT "(E)NCORE (M)ENU DISQUETTE (F)IN ";:
       GET R$
150  IF R$ = "M" THEN PRINT D$"RUN  /R4/MENU"
155  IF R$ = "E" THEN 125
165  IF R$ = "F" THEN HOME : END
170  GOTO 145
175  CALL - 198: POKE 49168,0: WAIT 49152,128: POKE 49168,0:
       RETURN
```

```
300 :  A2  5F       LDX  £$5F
302 :  A9  20       LDA  £$20
304 :  85  E6       STA  $E6
306 :  8A           TXA
307 :  20  2F  03   JSR  $032F
30A :  A5  26       LDA  $26
30C :  85  06       STA  $06
30E :  A5  27       LDA  $27
310 :  85  07       STA  $07
312 :  86  08       STX  $08
314 :  A9  BF       LDA  £$BF
316 :  38           SEC
317 :  E5  08       SBC  $08
319 :  20  2F  03   JSR  $032F
```

On ne traite que la moitié de l'écran.
Pour HGR ($40 pour HGR2).

L'adresse de la ligne comprise dans la partie haute de l'écran est stockée en $6-7.

Numéro de ligne ajusté.
L'adresse de la ligne comprise dans la partie basse de l'écran restera en $26-27. SEC est superflu.

```
31C :  AØ 27     LDY  £$27
31E :  B1 Ø6     LDA  ($Ø6),Y
32Ø :  48        PHA
321 :  B1 26     LDA  ($26),Y
323 :  91 Ø6     STA  ($Ø6),Y
325 :  68        PLA
326 :  91 26     STA  ($26),Y
328 :  88        DEY
329 :  1Ø F3     BPL  $Ø31E
32B :  CA        DEX
32C :  1Ø D8     BPL  $Ø3Ø6
32E :  6Ø        RTS
```

La ligne A prend la place de la ligne B et vice versa.

X = X - 1.

BASIC.

HPOSN

```
32F :  48        PHA
33Ø :  29 CØ     AND  £$CØ
332 :  85 26     STA  $26
334 :  4A        LSR
335 :  4A        LSR
336 :  Ø5 26     ORA  $26
338 :  85 26     STA  $26
33A :  68        PLA
33B :  85 27     STA  $27
33D :  ØA        ASL
33E :  ØA        ASL
33F :  ØA        ASL
34Ø :  26 27     ROL  $27
342 :  ØA        ASL
343 :  26 27     ROL  $27
345 :  ØA        ASL
346 :  66 26     ROR  $26
348 :  A5 27     LDA  $27
34A :  29 1F     AND  £$1F
34C :  Ø5 E6     ORA  $E6
34E :  85 27     STA  $27
35Ø :  6Ø        RTS
```

HGR.RTN1

L'IMAGE HGR est renversée et le texte n'est donc pas à l'envers, comme c'est le cas dans HGR.RTN. Une seule ligne change par rapport à **HGR.RBAS** :

HGR.RTN1B

```
1Ø5 PRINT D$"BLOAD/R4/HGR/HGR.RTN1"
```

```
3ØØ :  A2 5F      LDX  £$5F
3Ø2 :  A9 2Ø      LDA  £$2Ø
3Ø4 :  85 E6      STA  $E6
3Ø6 :  8A         TXA
3Ø7 :  2Ø 3C Ø3   JSR  $Ø33C
3ØA :  A5 26      LDA  $26
3ØC :  85 Ø6      STA  $Ø6
```

Voir HGR.RTN

(suite page 52)

```
30E :  A5 27      LDA  $27
310 :  85 07      STA  $07
312 :  86 08      STX  $08
314 :  A9 BF      LDA  £$BF      Voir HGR.RTN.
316 :  38         SEC
317 :  E5 08      SBC  $08
319 :  20 3C 03   JSR  $033C
31C :  A0 27      LDY  £$27
31E :  B1 06      LDA  ($06),Y
320 :  48         PHA
321 :  B1 26      LDA  ($26),Y   Les valeurs lues
323 :  48         PHA            sont stockées sur la pile.
324 :  88         DEY
325 :  10 F7      BPL  $031E
327 :  A0 27      LDY  £$27
329 :  68         PLA
32A :  20 5E 03   JSR  $035E
32D :  91 06      STA  ($06),Y   On retrouve sur la pile les valeurs
32F :  68         PLA            lues plus haut, mais il faut "retour-
330 :  20 5E 03   JSR  $035E     ner" chaque octet.
333 :  91 26      STA  ($26),Y
335 :  88         DEY
336 :  10 F1      BPL  $0329
338 :  CA         DEX
339 :  10 CB      BPL  $0306     Encore.
33B :  60         RTS            Terminé.
```

HPOSN

```
33C :  48         PHA
33D :  29 C0      AND  £$C0
33F :  85 26      STA  $26
341 :  4A         LSR
342 :  4A         LSR
343 :  05 26      ORA  $26
345 :  85 26      STA  $26
347 :  68         PLA
348 :  85 27      STA  $27
34A :  0A         ASL
34B :  0A         ASL
34C :  0A         ASL
34D :  26 27      ROL  $27
34F :  0A         ASL
350 :  26 27      ROL  $27
352 :  0A         ASL
353 :  66 26      ROR  $26
355 :  A5 27      LDA  $27
357 :  29 1F      AND  £$1F
359 :  05 E6      ORA  $E6
35B :  85 27      STA  $27
35D :  60         RTS
```

OCTET RETOURNÉ

```
35E :  5A         PHY            Y sur la pile.
35F :  A0 08      LDY  £$08
361 :  0A         ASL
362 :  66 18      ROR  $18       Où 01100111 devient 01110011.
364 :  88         DEY
365 :  10 FA      BPL  $0361
367 :  7A         PLY
368 :  A5 18      LDA  $18
36A :  60         RTS
```

— VERSION PLUS LONGUE ET PLUS RAPIDE (HGR.RTN1) —

```
35E :  0A          ASL
35F :  0A          ASL
360 :  66  18      ROR  $18
362 :  0A          ASL
363 :  66  18      ROR  $18
365 :  0A          ASL
366 :  66  18      ROR  $18
368 :  0A          ASL
369 :  66  18      ROR  $18
36B :  0A          ASL
36C :  66  18      ROR  $18
36E :  0A          ASL
36F :  66  18      ROR  $18
371 :  0A          ASL
372 :  66  18      ROR  $18
374 :  A5  18      LDA  $18
376 :  4A          LSR
377 :  60          RTS
```

Le nombre de cycles est moins élevé d'où une plus grande rapidité d'exécution.
La version courte est ralentie par PHY et PLY (que ne connaît d'ailleurs pas le 6502), mais aussi par la boucle (LDY et surtout DEY et BPL) qui pénalise la routine.

PRT.USVAR

Une autre version d'un PRINT USING élémentaire, mais avec possibilité d'accepter un ou plusieurs chiffres après la virgule.

```
10 FOR I = 768 TO 831: READ R: POKE I,R: NEXT
15 DATA  32,245,230,134,24,32,245,230,134,9,32,190,222,32,227,223,160,2,177,131,
   153,6,0,136,16,248,200,196,6,240,23,177,7,201,46,208,245,169,44,145,7,152,101,24,
   197,6,16,6,90,160,0,145,131,122,165,9,132,9,56,229,9,133,36,96
20 TEXT : NORMAL : PRINT CHR$ (21): HOME
25 U = 2:H = 9:U1 = 3:H1 = 24
30 FOR I = 1 TO 12: READ R:A$ = STR$ (R): CALL 768,U,H,A$: PRINT A$;: READ
    R:A$ = STR$ (R): CALL 768,U1,H1,A$: PRINT A$: NEXT
35 VTAB 22: PRINT "(M)ENU DE DISQUETTE ";: GET R$: IF R$ = "M" OR R$ = "m"
    THEN PRINT CHR$ (4)"RUN MENU"
40 HOME : END
45 DATA 0.23,123,1230.1,235.10,78649.80,.17,1,75.2,12345.2575,.2345,10,17.0101,
   101
50 DATA 0.234,56789.12,2,490,535.60,23.4555,286.1234,12.05,9999.999,9009.0909,
   108.99999
```

(Suite page 54).

```
300 : 20 F5 E6   JSR  $E6F5   ┐ Nombre de chiffres après la vir-
303 : 86 18      STX  $18     ┘ gule en $18.
305 : 20 F5 E6   JSR  $E6F5   ┐ HTAB en $9.
308 : 86 09      STX  $09     ┘
30A : 20 BE DE   JSR  $DEBE   ┐
30D : 20 E3 DF   JSR  $DFE3   │ Le nombre a préalablement été
310 : A0 02      LDY  £$02    │ transformé en variable alphanu-
312 : B1 83      LDA  ($83),Y │ mérique dans le programme en
314 : 99 06 00   STA  $0006,Y │ Basic (STR$). Son adresse est en
317 : 88         DEY          │ $7.8 et sa longueur en $6.
318 : 10 F8      BPL  $0312   ┘
```

```
31A : C8         INY          Y = Y + 1 (0 au départ).
31B : C4 06      CPY  $06     ┐ Si Y est égal à la longueur, c'est
31D : F0 17      BEQ  $0336   ┘ fini !
31F : B1 07      LDA  ($07),Y Lecture d'un chiffre.
321 : C9 2E      CMP  £$2E    ┐ Si ce n'est pas le point, voir plus
323 : D0 F5      BNE  $031A   ┘ loin.
325 : A9 2C      LDA  £$2C    ┐ Sinon, on le transforme en
327 : 91 07      STA  ($07),Y ┘ virgule.
329 : 98         TYA          ┐
32A : 65 18      ADC  $18     │ Comparaison de Y + nombre de
32C : C5 06      CMP  $06     │ décimales avec longueur. Saut
32E : 10 06      BPL  $0336   ┘ éventuel.
```

```
330 : 5A         PHY          Y empilé.
331 : A0 00      LDY  £$00    ┐ Nouvelle longueur de la variable
333 : 91 83      STA  ($83),Y ┘ installée.
335 : 7A         PLY          Y récupéré.
```

```
336 : A5 09      LDA  $09     ┐
338 : 84 09      STY  $09     │
33A : 38         SEC          │ On détermine ici le HTAB réel.
33B : E5 09      SBC  $09     │ (Valable en 40 colonnes).
33D : 85 24      STA  $24     ┘
33F : 60         RTS          Basic. ■
```

PRBLNK $F948

```
F948 : A2 03     LDX  £$03
F94A : A9 A0     LDA  £$A0
F94C : 20 ED FD  JSR  $FDED
F94F : CA        DEX
F950 : D0 F8     BNE  $F94A
F952 : 60        RTS
```

Comme on le voit, cette routine du moniteur affiche 3 espaces (X=3), mais il est possible d'afficher n'importe qu'elle autre valeur de X en se branchant non plus sur $F948, mais sur $F94A (PRBL2). On peut aussi afficher X espaces à la suite d'un caractère choisi, mais en se branchant cette fois sur $F94C (PRBL3), comme ceci :

```
300 : A9 AA      LDA £$AA
302 : A2 FF      LDX £$FF
304 : 4C 4C F9   JMP $F94C
```

Où l'on affiche l'astérisque + 255 espaces.

PRT.USPIL

Intéressante utilisation de FOUT (qui convertit la valeur de FAC en chaîne de chiffres, comme on le verra en lisant les explications de la routine).

```
10  FOR I = 768 TO 825: READ R: POKE I,R: NEXT
15  DATA     32,76,231,134,6,32,190,222,32,103,221,32,52,237,160,0,185,0,1,240,7,
    201,46,240,3,200,208,244,165,6,132,6,229,6,133,36,160,0,185,0,1,240,14,
    73,128,201,174,208,2,169,172,32,237,253,200,208,237,96
20  TEXT : NORMAL : PRINT CHR$ (21): HOME
25  FOR I = 0 TO 100 STEP 5:A = I *  133.275: GOSUB 40: NEXT
30  VTAB 23: PRINT "(M)ENU DE DISQUETTE ";: GET R$: IF R$ = "M" OR R$ = "m"
     THEN PRINT CHR$ (4)"RUN MENU"
35  END
40  CALL 768,10,A:A = A *  2: CALL 768,28,A: PRINT : RETURN
```

```
300 :  20 4C E7     JSR   $E74C      ┐ HTAB installé en $6.
303 :  86 06        STX   $06        ┘
305 :  20 BE DE     JSR   $DEBE      CHKCOM teste la virgule.
308 :  20 67 DD     JSR   $DD67      FRMNUM évalue la formule (rangement dans FAC).
30B :  20 34 ED     JSR   $ED34      FOUT convertit FAC en chaîne de chiffres.
30E :  A0 00        LDY   £$00       ┐
310 :  B9 00 01     LDA   $0100,Y    │ Cette chaîne est récupérée à la base de
313 :  F0 07        BEQ   $031C      │ la pile, à partir de $100.
315 :  C9 2E        CMP   £$2E       │ On lit chaque octet. S'il s'agit d'un 0, on
317 :  F0 03        BEQ   $031C      │ a terminé. S'il s'agit de $2E (le point), on
319 :  C8           INY              │ passe aussi à la suite.
31A :  D0 F4        BNE   $0310      ┘
-------------------------------------------------------------------------------
31C :  A5 06        LDA   $06        ┐
31E :  84 06        STY   $06        │ HTAB ajusté.
320 :  E5 06        SBC   $06        │
322 :  85 24        STA   $24        ┘
-------------------------------------------------------------------------------
324 :  A0 00        LDY   £$00       ┐
326 :  B9 00 01     LDA   $0100,Y    │
329 :  F0 0E        BEQ   $0339      │ Affichage de chaque chiffre.
32B :  49 80        EOR   £$80       │ EOR force le bit 7 à 1 pour COUT
32D :  C9 AE        CMP   £$AE       │ ($FDED). Si on rencontre le point
32F :  D0 02        BNE   $0333      │ ($AE), on le transforme en virgule
331 :  A9 AC        LDA   £$AC       │ ($AC). La valeur 0 dans A termine la
333 :  20 ED FD     JSR   $FDED      │ séquence.
336 :  C8           INY              │
337 :  D0 ED        BNE   $0326      ┘
339 :  60           RTS              BASIC.                                  ■
```

PRT.USPIL1

Même routine que, PRT.USPIL, mais permettant d'accepter un nombre déterminé de décimales.

```
10  FOR I = 768 TO 839: READ R: POKE I,R: NEXT
15  DATA    32,76,231,232,134,7,32,76,231,134,6,32,190,222,32,103,221,32,52,237,
    160,0,185,0,1,240,7,201,46,240,3,200,208,244,165,6,132,6
20  DATA    229,6,133,36,166,7,160,0,185,0,1,240,20,73,128,201,174,208,3,202,169,
    172,32,237,253,200,228,7,240,234,202,16,231,96
25  U0 = 1:U1 = 2
30  TEXT : NORMAL : PRINT CHR$ (21): HOME
35  FOR I = 0 TO 100 STEP 5:A = I * 133.275: GOSUB 50: NEXT
40  VTAB 23: PRINT "(M)ENU DE DISQUETTE ";: GET R$: IF R$ = "M" OR R$ = "m"
     THEN PRINT CHR$ (4)"RUN MENU"
45  END
50  CALL 768,U0,10,A:A = A * 2: CALL 768,U1,28,A: PRINT : RETURN
```

```
300 :  20 4C E7    JSR   $E74C
303 :  E8          INX
304 :  86 07       STX   $07
```

Nombre de décimales + 1 en $7.

```
306 :  20 4C E7    JSR   $E74C
309 :  86 06       STX   $06
30B :  20 BE DE    JSR   $DEBE
30E :  20 67 DD    JSR   $DD67
311 :  20 34 ED    JSR   $ED34
314 :  A0 00       LDY   £$00
316 :  B9 00 01    LDA   $0100,Y
319 :  F0 07       BEQ   $0322
```

```
31B :  C9 2E       CMP   £$2E
31D :  F0 03       BEQ   $0322
31F :  C8          INY
320 :  D0 F4       BNE   $0316
322 :  A5 06       LDA   $06
324 :  84 06       STY   $06
326 :  E5 06       SBC   $06
328 :  85 24       STA   $24
```

```
32A :  A6 07       LDX   $07
32C :  A0 00       LDY   £$00
32E :  B9 00 01    LDA   $0100,Y
331 :  F0 14       BEQ   $0347
333 :  49 80       EOR   £$80
335 :  C9 AE       CMP   £$AE
337 :  D0 03       BNE   $033C
339 :  CA          DEX
33A :  A9 AC       LDA   £$AC
33C :  20 ED FD    JSR   $FDED
33F :  C8          INY
340 :  E4 07       CPX   $07
342 :  F0 EA       BEQ   $032E
344 :  CA          DEX
345 :  10 E7       BPL   $032E
347 :  60          RTS
```

Nombre de décimales dans X.

Affichage similaire à celui expliqué dans PRT.USPIL, mais on tient compte de la valeur de X (nombre de décimales).

■

DATA-SUITE

Voici un moyen simple et rapide pour afficher les données en DATA sans passer par une boucle en Basic. Attention ! comme il s'agit réellement d'une routine élémentaire, il faut absolument initialiser le système à partir du Basic par une instruction similaire à celle de la ligne 20 (READ R$). La première donnée est bidon (ici : 0).

```
10 TEXT : HOME : PRINT CHR$ (4)"BLOAD DS.LM"
20 READ R$
30 CALL 768: PRINT
40 VTAB 22: PRINT "PRESSER (M) POUR MENU DE DISQUETTE ";: GET R$:
   PRINT
50 IF R$ = "M" OR R$ = "m" THEN PRINT CHR$ (4)"RUN MENU"
60 HOME : END
70 DATA  0,JANVIER,FEVRIER,MARS,AVRIL,MAI,JUIN,JUILLET,AOUT,
   SEPTEMBRE,OCTOBRE,NOVEMBRE,DECEMBRE
```

300 :	A0 00		LDY	£$00	Y à 0 pour adressage indirect.
302 :	E6 7F		INC	$7F	Après READ R$, $7F-80 contient l'adresse où se trouve la virgule suivant le 0.
304 :	D0 02		BNE	$0308	
306 :	E6 80		INC	$80	
308 :	B1 7F		LDA	($7F),Y	Lecture du caractère.
30A :	F0 10		BEQ	$031C	Si c'est un zéro, lecture épuisée.
30C :	C9 2C		CMP	£$2C	Si c'est une virgule, retour chariot. Sinon on affiche le caractère.
30E :	D0 05		BNE	$0315	
310 :	20 62	FC	JSR	$FC62	
313 :	80 EB		BRA	$0300	Retour.
315 :	09 80		ORA	£$80	Bit 7 à 1. Affichage et en place pour une autre lecture.
317 :	20 ED	FD	JSR	$FDED	
31A :	80 E4		BRA	$0300	
31C :	60		RTS		Retour au Basic.

Vous avez bien sûr remarqué que ce micro-programme n'autorise pas l'affichage de plusieurs lignes de DATA. La raison est évidente : on se contente ici de court-circuiter le Basic. Vous en aurez la preuve en ajoutant une ligne 35 :

35 READ R$: PRINT R$

Que croyez-vous qu'elle affichera ? ■

DATA-S1

Même principe que DATA.SUITE, mais le programme affiche l'adresse décimale de chaque donnée et traite toutes les lignes de DATA.

```
10  TEXT : HOME : PRINT CHR$ (4)"BLOAD DS.LM1"
20  DATA PREMIERE LIGNE
30  CALL 768: PRINT
40  VTAB 22: PRINT "PRESSER (M) POUR MENU DE DISQUETTE ";: GET R$:
    PRINT
50  IF R$ = "M" OR R$ = "m" THEN PRINT CHR$ (4)"RUN MENU"
60  HOME : END
70  DATA  JANVIER,FEVRIER,MARS,AVRIL,MAI,JUIN,JUILLET,AOUT,
    SEPTEMBRE,OCTOBRE,NOVEMBRE,DECEMBRE
75  DATA PREMIERE SUITE,LIGNE 75
80  DATA DEUXIEME SUITE,LIGNE 80
85  DATA TROISIEME SUITE,LIGNE 85
90  PRINT : REM  LIGNE BIDON
95  DATA ET ON TERMINE LIGNE 95
```

```
300 :   A5 67        LDA   $67
302 :   85 7F        STA   $7F
304 :   A5 68        LDA   $68
306 :   85 80        STA   $80
308 :   80 31        BRA   $033B
30A :   A4 80        LDY   $80
30C :   A6 7F        LDX   $7F
30E :   E8           INX
30F :   D0 01        BNE   $0312
311 :   C8           INY
312 :   98           TYA
313 :   20 24 ED     JSR   $ED24
316 :   20 48 F9     JSR   $F948
319 :   A0 00        LDY   £$00
31B :   E6 7F        INC   $7F
31D :   D0 02        BNE   $0321
31F :   E6 80        INC   $80
321 :   B1 7F        LDA   ($7F),Y
```

L'adresse de début du programme Applesoft est placée en $7F-80... comme le fait RESTORE.

Saut.

On affiche l'adresse décimale du premier octet de chaque donnée. C'est LINPTR ($ED24) qui s'en charge. Ensuite PRBLNK ($F948) envoie 3 espaces.

Registre Y à 0.

Adresse incrémentée. Si on obtient 0, la partie haute est aussi incrémentée.

Lecture.

```
323 : F0 0B      BEQ  $0330    Si c'est un 0, saut.
325 : C9 2C      CMP  £$2C     Si c'est une virgule, saut également.
327 : F0 07      BEQ  $0330
329 : 09 80      ORA  £$80     Affichage du caractère après mise
32B : 20 ED FD   JSR  $FDED    à 0 du bit 7.
32E : 80 E9      BRA  $0319    Et on continue...
330 : 48         PHA           Valeur de A empilée.
331 : A9 8D      LDA  £$8D     On envoie un RETURN (nécessaire
333 : 20 ED FD   JSR  $FDED    en 80 colonnes).
336 : 68         PLA           Récupération sur la pile.
337 : C9 2C      CMP  £$2C     Si c'est une virgule, on va afficher
339 : F0 CF      BEQ  $030A    l'adresse du DATA suivant*.
33B : E6 7F      INC  $7F      Adresse incrémentée
33D : D0 02      BNE  $0341    comme plus haut.
33F : E6 80      INC  $80
341 : A0 00      LDY  £$00     Registre Y à 0.
343 : B1 7F      LDA  ($7F),Y  Lecture de l'octet.
345 : F0 06      BEQ  $034D    Si c'est un 0, saut.
347 : C9 83      CMP  £$83     Si c'est le token de DATA, c'est
349 : F0 BF      BEQ  $030A    parti mon kiki !
34B : D0 EE      BNE  $033B    Sinon on va incrémenter de nouveau.
34D : A5 80      LDA  $80
34F : C5 6A      CMP  $6A
351 : 90 E8      BCC  $033B    Si on n'est pas en fin de pro-
353 : A5 7F      LDA  $7F      gramme Applesoft, on continue.
355 : C5 69      CMP  $69      Sinon, retour au Basic.
357 : 90 E2      BCC  $033B
359 : 60         RTS
```

** Si une ligne DATA se termine par une virgule, on est coincé !*

ADRCHG

Programme de démonstration sur votre disquette

Vous n'êtes pas obligé de charger un fichier binaire à partir de son premier octet. Vous pouvez escamoter B octets.

Ainsi, un **BRUN FIC,B$12** va installer FIC à son adresse habituelle, mais en négligeant les $12 (18) premiers octets.

Même effet avec **BLOAD FIC,B$12**.

En Basic, c'est un peu la même chose avec **RUN PROG,à200**... qui charge la totalité de PROG, mais le lance à partir de la ligne 200. Attention ! ne fonctionne pas avec le dash (-). ■

MÉLANGEUR

Comment mélanger aléatoirement les N mots d'un fichier binaire (voir *SAISIE* — dans nos *ROUTINES LM* — et *MULTIE.SAISIE* — dans nos *NOUVELLES ROUTINES ?*
Par exemple en utilisant **MÉLANGEUR**.

PRÉCAUTION :

Votre fichier devra impérativement se terminer par L×2 zéros (L = longueur d'un mot, soit 8 zéros pour des mots de 4 lettres).

MEL.BAS

```
100  D$ = CHR$ (4): PRINT D$"BLOAD MELANGEUR"
110  PRINT D$"PR£3": PRINT : HOME
120  PRINT D$"CATALOG"
130  PRINT : INPUT "TITRE A TRAITER ? ";T$: IF T$ = "" THEN
     END
140  INPUT "LONGUEUR DES MOTS ? ";L: IF NOT L THEN END
150  PRINT D$"BLOAD"T$
160  CALL 768,L
170  A = L + 1 + ( PEEK (8) + PEEK (9) * 256)
180  LF = 1 + L * 2 + ( PEEK (25) + PEEK (26) * 256) - A
190  PRINT D$"BSAVE";T$ + "M";",A";A;",L";LF
200  PRINT D$"CATALOG"
210  PRINT
220  VTAB 23: PRINT "(M)ENU DE DISQUETTE (F)IN ";: GET
     R$: VTAB 22: PRINT
230  IF R$ = "M" OR R$ = "m" THEN PRINT D$"RUN
     MENU,S6"
240  IF R$ < > "F" AND R$ < > "f" THEN 220
250  HOME
```

MEL.ESSAI

```
100  D$ = CHR$ (4): PRINT D$"BLOAD
     MELANGEUR"
110  PRINT D$"PR£3": PRINT : HOME
120  PRINT D$"BLOAD ES5"
130  L = 5
140  HOME
150  CALL 768,L
160  A = L + 1 + ( PEEK (8) + PEEK
     (9) * 256)
170  LF = 1 + ( PEEK (25) + PEEK
     (26) * 256)
180  FOR I = A TO A + L - 1: M$ = M$
     + CHR$ ( PEEK (I)): NEXT
190  N = N + 1:M$(N) = M$:M$ = "":A
     = A + L: IF A < LF THEN 180
200  FOR I = 1 TO N: PRINT M$(I),:
     NEXT
210  PRINT : PRINT
220  N = 0:S = S + 1: IF S < 6 THEN
     150
230  VTAB 23: PRINT "(M)ENU DE DIS
     QUETTE (F)IN ";: GET R$: VTAB 22:
     PRINT
240  IF R$ = "M" OR R$ = "m" THEN
     PRINT D$"RUN MENU,S6"
250  IF R$ < > "F" AND R$ < > "f"
     THEN 230
260  HOME
```

MÉLANGEUR

POINTEURS : **$18** = Longueur d'un mot — **$08-09** = Adresse de base du fichier (moins la longueur d'un mot et moins un octet) — **$06-07** = Adresse du mot choisi aléatoirement — **$19-1A** = Adresse du mot traité séquentiellement — **$1B-1C** = Nombre de mots moins deux.

LONGUEUR DES MOTS

```
300 : 20 F5 E6   JSR E6F5
303 : 86 18      STX 18
```

GETBYTC saute un caractère, puis appelle GETBYT et CONINT. La valeur envoyée par CALL est dans X.

ADRESSE DU FICHIER

```
305 : AD B9 BE   LDA BEB9
308 : 18         CLC
309 : E5 18      SBC 18
30B : 85 08      STA 08
30D : 85 19      STA 19
30F : AD BA BE   LDA BEBA
312 : E9 00      SBC #00
314 : 85 1A      STA 1A
316 : 85 09      STA 09
```

La partie basse est diminuée de la longueur d'un mot + 1 (on annule la retenue, comme pour une addition, ce qui diminue le résultat d'un octet).
Stockage en $8 et $19.

Voir la note à propos de $BEB9-BEBA (page 63).

La partie haute va en $9 et $1A.
Sous DOS 3.3, on utiliserait $AA72 et $AA73 au lieu de $BEB9-BEBA.

LONGUEUR DU FICHIER (réelle)

```
318 : AC C8 BE   LDY BEC8
31B : AD C9 BE   LDA BEC9
31E : 20 F2 E2   JSR E2F2
321 : 20 63 EB   JSR EB63
```

Longueur réelle du fichier dans Y et A. Sous DOS 3.3, remplacer $BEC8-BEC9 par $AA60-AA61.
GIVAYF rend flottant l'entier relatif de Y,A.
MOVAF transfère FAC dans ARG.

NOMBRE DE MOTS (moins 3)

```
324 : A4 18      LDY 18
326 : A9 00      LDA #00
328 : 20 F2 E2   JSR E2F2
32B : 20 69 EA   JSR EA69
32E : 20 52 E7   JSR E752
```

Longueur du mot dans Y. A = 0.
L'entier relatif de Y,A devient flottant.
FDIVT s'occupe de la division (résultat dans FAC).
GETADR transforme FAC en 2 octets (dans Y,A).

```
331 : 85 1C      STA 1C
333 : 98         TYA
334 : 38         SEC
335 : E9 03      SBC #03
337 : 85 1B      STA 1B
339 : B0 02      BCS 033D
33B : C6 1C      DEC 1C
```

Stockage de la partie haute dans $1C.
La partie basse passe dans A.
Nombre de mots diminué de 3 unités.
Stockage de la partie basse dans $1B.
Partie haute décrémentée si nécessaire.

ADRESSE DU MOT À TRAITER

```
33D : 18         CLC
33E : A5 18      LDA 18
340 : 65 19      ADC 19
342 : 85 19      STA 19
344 : 90 02      BCC 0348
346 : E6 1A      INC 1A
```

Annulation de la retenue.
Pointeurs de l'adresse de traitement majorés de la longueur d'un mot. La partie haute n'est incrémentée que s'il y a retenue.

(suite page 62)

NOMBRE ALÉATOIRE — MOT ALÉATOIRE

```
348 : A9 01       LDA £01     ┐ Entrée spéciale pour RND.
34A : 20 B1 EF    JSR EFB1    ┘
34D : 20 63 EB    JSR EB63    MOVAF transfère FAC dans ARG.
350 : A4 1B       LDY 1B      ┐ Nombre de mots (toujours moins 3) dans Y et A.
352 : A5 1C       LDA 1C      ┘
354 : 20 F2 E2    JSR E2F2    GIFAYF rend l'entier flottant.
357 : 20 82 E9    JSR E982    FMULTT se charge de multiplier l'argument par FAC.
35A : 20 63 EB    JSR EB63    MOVAF transfère FAC dans ARG.
35D : 20 52 E7    JSR E752    FAC transformé en entier sur 2 octets, dans Y et A.
360 : C8          INY         Y = Y + 1 pour éviter le 0.
361 : D0 01       BNE 0364    ┐ Si Y = 0, il faut incrémenter A (partie haute).
363 : 1A          INC         ┘ (Attention ! pas de "1A" sur le vieux 6502 !).
364 : 20 F2 E2    JSR E2F2    L'entier de Y, A devient (ou redevient ?) flottant.
367 : 20 63 EB    JSR EB63    Toujours MOVAF (FAC dans ARG).
```

ADRESSE DU MOT ALÉATOIREMENT CHOISI

```
36A : A4 18       LDY 18      ┐ Longueur du mot dans Y,A.
36C : A9 00       LDA £00     ┘
36E : 20 01 E3    JSR E301    SGNFLT rend l'entier de Y flottant.
371 : 20 82 E9    JSR E982    FMULTT multiplie ARG par FAC.
374 : 20 52 E7    JSR E752    GETADR transforme FAC en entier sur 2 octets (Y,A).
377 : 48          PHA         A est empilé.
378 : 98          TYA         Y passe dans A (partie basse de l'adresse).
379 : 18          CLC         ┐ Addition et mise à jour de l'adresse du mot choisi
37A : 65 08       ADC 08      │ aléatoirement.
37C : 85 06       STA 06      ┘
37E : 68          PLA         A est récupéré.
37F : 65 09       ADC 09      ┐ Même opération sur la partie haute de l'adresse.
381 : 85 07       STA 07      ┘
```

LECTURE D'UN MOT (ordre séquentiel)

```
383 : A4 18       LDY 18      Longueur du mot dans Y.
385 : B1 19       LDA (19),Y  Lecture d'un caractère.
387 : F0 12       BEQ 039B    Si c'est un zéro, le mélange est terminé.
389 : 48          PHA         Octet empilé.
38A : 88          DEY         Y = Y - 1 (jusqu'à 1 inclus).
38B : D0 F8       BNE 0385    Quand Y = 0, on arrête.
```

PERMUTATION

```
38D : C8          INY         Y = Y + 1 (au départ, Y = 0 + 1).
38E : B1 06       LDA (06),Y  Lecture d'un caractère du mot choisi aléatoirement.
390 : 91 19       STA (19),Y  Ecriture de l'octet du mot traité.
392 : 68          PLA         Récupération sur la pile.
393 : 91 06       STA (06),Y  Remplacement.
395 : C4 18       CPY 18      ┐ Si Y est égal à la longueur du mot, on passe au suivant.
397 : D0 F4       BNE 038D    │ Sinon, on continue la permutation.
399 : 80 A2       BRA 033D    ┘
39B : 60          RTS         Retour au Basic.
```

BSAVE MÉLANGEUR,A$300,L$9C

LPRODOS

C'est entendu, il suffit de taper CATALOG pour connaître la longueur d'un fichier ProDOS, mais on peut aussi l'obtenir comme ceci, par CALL 768 ou CALL 783 (plus efficace).

VERSION A

```
300 : AC C8 BE   LDY  $BEC8   ┐ Longueur du fichier chargé dans Y et A.
303 : AD C9 BE   LDA  $BEC9   ┘
306 : 20 F2 E2   JSR  $E2F2     GIVAYF rend flottant l'entier relatif dans Y,A.
309 : 20 34 ED   JSR  $ED34     FOUT convertit la valeur de FAC en chaîne de chiffres.
30C : 4C 3A DB   JMP  $DB3A     STROUT affiche cette chaîne (pointée par Y,A).
```

VERSION B

```
30F : AE C8 BE   LDX  $BEC8   ┐ Longueur dans X et A.
312 : AD C9 BE   LDA  $BEC9   ┘
315 : DA         PHX          ┐ X et A empilés.
316 : 48         PHA          ┘
317 : 20 24 ED   JSR  $ED24     LINPTR affiche directement la valeur décimale de X,A.
31A : 20 48 F9   JSR  $F948     PRBLNK affiche 3 espaces.
31D : 68         PLA          ┐ A et X chargés.
31E : FA         PLX          ┘
31F : 4C 41 F9   JMP  $F941     PRNTAX affiche les registres A et X (hexa). ■
```

LMPRO

Routine tirée de MELANGEUR. La longueur du mot doit être poquée dans $18 (POKE 24,L).

```
300 : AC C8 BE   LDY  $BEC8   ┐ Longueur du fichier dans Y et A.
303 : AD C9 BE   LDA  $BEC9   ┘
306 : 20 F2 E2   JSR  $E2F2     GIVAYF rend l'entier flottant (Y,A).
309 : 20 63 EB   JSR  $EB63     MOVAF transfère FAC dans ARG.
30C : A4 18      LDY  $18       Y va contenir la longueur d'un mot (POKE 24,L).
30E : 20 01 E3   JSR  $E301     SGNFLT rend flottant l'entier de Y.
311 : 20 69 EA   JSR  $EA69     FDNT se charge de diviser (ARG/FAC).
314 : 20 34 ED   JSR  $ED34     FOUT convertit FAC en chaîne de chiffres.
317 : 4C 3A DB   JMP  $DB3A     STROUT affiche cette chaîne. ■
```

À PROPOS DE $BEB9-BEBA

Sous ProDOS (voir la partie *Adresse du Fichier*), on lit l'adresse en question en FIAUX10 ($BEB9-BEBA), mais ce renseignement sera inexact si l'on a chargé le fichier à une adresse différente de celle de l'enregistrement. Il est alors préférable de lire l'adresse d'un fichier chargé dans :

— RWDATA $BED7-BED8 Pointeur pour lecture-écriture de données.
— RWCOUNT $BED9-BEBA Fournit le nombre de bytes, comme $BEC8-BEC9.

DIVISION

On nous demande souvent comment, à partir du Basic, transmettre deux nombres à une routine LM qui effectuera la division...

```
100  TEXT : NORMAL :D$ = CHR$ (4): PRINT D$"PR£3": PRINT
110  FOR I = 768 TO 976: READ R: POKE I,R: IF R <> 0 THEN NEXT
120  HOME : PRINT "D I V I S I O N": PRINT "_____ "
130  PRINT : VTAB 12: CALL - 958
140  INPUT "Nombre: ";R$:N = VAL (R$): IF NOT N THEN 190
150  INPUT "Diviseur: ";R$:D = VAL (R$): IF NOT D THEN 130
160  CALL 768,N,D: PRINT
170  PRINT : PRINT N / D
180  CALL - 198: POKE 49168,0: WAIT 49152,128: POKE 49168,0: IF
     PEEK (49152) <> 27 THEN 130
190  VTAB 22: PRINT : PRINT "(E)NCORE (M)ENU DE DISQUETTE
     (A)PPLESOFT ";:  GET R$
200  IF R$ = "E" OR R$ = "e" THEN 130
210  IF R$ = "M" OR R$ = "m" THEN PRINT D$"RUN MENU"
220  IF R$ = "A" OR R$ = "a" THE HOME : END
230  GOTO 190
240  DATA  32,190,222,32,123,221,32,99,235,32,190,222,32,123,221,32,
     105,234,32,52,237,76,58,219,0
```

■

DIVIDENDE

```
300:  20 BE DE    JSR  $DEBE
303:  20 7B DD    JSR  $DD7B
306:  20 63 EB    JSR  $EB63
```

CHKCOM : Teste la virgule en TXTPTR.
FRMEVL : Evalue la formule et place le résultat en FAC.
MOVAF : FAC dans ARG.

DIVISEUR

```
309:  20 BE DE    JSR  $DEBE
30C:  20 7B DD    JSR  $DD7B
30F:  20 69 EA    JSR  $EA69
312:  20 34 ED    JSR  $ED34
315:  4C 3A DB    JMP  $DB3A
```

Même tabac.
FDIVT : ARG / FAC (résultat dans FAC).
FOUT : Convertit la valeur de FAC en chaîne de chiffres.
STROUT : Affiche la chaîne. ■

QUATRE OPÉRATIONS

Comment transmettre des données à une routine LM de calcul ? Voici une solution simple, facile à comprendre.

REMARQUE : On envoie X = Ø pour la soustraction, X = 1 pour l'addition, 2 pour la multiplication et 3 pour la division. On peut utiliser × à la place de * (multiplication) et : à la place de / (division).

```
100  TEXT : NORMAL :D$ = CHR$ (4): PRINT D$"PR£3": PRINT
110  FOR I = 768 TO 976: READ R: POKE I,R: IF R <> Ø THEN NEXT
120  HOME : PRINT "Q U A T R E  O P E R A T I O N S ":
          PRINT "_____"
130  PRINT : VTAB 12: CALL - 958
140  PRINT "Opération (N1 - N2 ou N1 + N2 ou N1 *  N2 ou N1 / N2)"
150  VTAB 14: INPUT "N1 —» ";R$:N1 = VAL (R$): IF NOT N1 THEN 230
160  VTAB 15: PRINT "SIGNE —» ";: GET R$: PRINT R$
170  FOR X = Ø TO 3: IF R$ <> MID$ ("-+×:",X + 1,1) THEN NEXT
180  IF X <> 4 THEN 200
190  FOR X = Ø TO 3: IF R$ <> MID$ ("-+ * /",X + 1,1) THEN NEXT : IF X
       = 4 THEN 160
200  VTAB 16: INPUT "N2 —» ";R$:N2 = VAL (R$): IF NOT N2 THEN 150
210  CALL 768,X,N1,N2
220  CALL - 198: POKE 49168,Ø: WAIT 49152,128: POKE 49168,Ø: IF PEEK
       (49152) <> 27 THEN 130
230  VTAB 22: PRINT : PRINT "(E)NCORE (M)ENU DE DISQUETTE (A)PPLE
     SOFT ";: GET R$
240  IF R$ = "E" OR R$ = "e" THEN 130
250  IF R$ = "M" OR R$ = "m" THEN PRINT D$"RUN MENU"
260  IF R$ = "A" OR R$ = "a" THEN HOME : END
270  GOTO 230
280  DATA  32,245,230,169,231,160,170,224,1,144,22,224,2,176,4,160,193,
     128,14,224,3,176,6,169,233,160,130,208,4,169,234,160,105,140,55,3,
     141,56,3,32,190,222,32,123,221,32,99,235,32,190,222,32,123,221,32,
     105,234,32,52,237,76,58,219,Ø
```

(suite page 66)

QUELLE OPÉRATION ?

```
300 :  20 F5 E6    JSR  $E6F5
```
GETBYTC saute un caractère et appelle GETBYT, puis CONINT. Le résultat est dans X.

```
303 :  A9 E7       LDA  £$E7
305 :  A0 AA       LDY  £$AA
```
$E7AA (FSUBT) : soustraction dans A et Y.

```
307 :  E0 01       CPX  £$01
309 :  90 16       BCC  $0321
```
Si X = 0, c'est bien la soustraction : saut.

```
30B :  E0 02       CPX  £$02
30D :  B0 04       BCS  $0313
```
Si X supérieur ou égal à 2, voir plus loin...

```
30F :  A0 C1       LDY  £$C1
311 :  80 0E       BRA  $0321
```
X = 1, c'est l'addition. $C1 dans Y pour FADDT ($E7C1).

```
313 :  E0 03       CPX  £$03
315 :  B0 06       BCS  $031D
```
Si X supérieur ou égal à 3, voir plus loin...

```
317 :  A9 E9       LDA  £$E9
319 :  A0 82       LDY  £$82
```
$E982 (FMULTT) : adresse de la routine de multiplication dans A et Y.

```
31B :  D0 04       BNE  $0321
```
Saut.

```
31D :  A9 EA       LDA  £$EA
31F :  A0 69       LDY  £$69
```
$EA69 (FDIVT) : adresse de la routine de division.

```
321 :  8C 37 03    STY  $0337
324 :  8D 38 03    STA  $0338
```
Mise en place de l'adresse de la routine utilisée.

```
327 :  20 BE DE    JSR  $DEBE
```
CHKCOM teste la virgule.

```
32A :  20 7B DD    JSR  $DD7B
```
FRMEVL : évalue la formule. Résultat dans FAC.

```
32D :  20 63 EB    JSR  $EB63
```
MOVAF : FAC dans ARG.

```
330 :  20 BE DE    JSR  $DEBE
333 :  20 7B DD    JSR  $DD7B
```
Même processus que ci-dessus pour N2.

```
336 :  20 82 E9    JSR  $E982
```
Routine choisie (adresse variable).

```
339 :  20 34 ED    JSR  $ED34
```
FOUT convertit la valeur de FAC en chaîne de chiffres.

```
33C :  4C 3A DB    JMP  $DB3A
```
STROUT affiche cette chaîne. ■

RND 0 OU 1 ?

Comment obtenir (aléatoirement bien sûr) une série de 0 ou de 1 ?

```
300 :  A9 01       LDA  £$01
302 :  20 B1 EF    JSR  $EFB1
```
Entrée spéciale de RND.

```
305 :  20 63 EB    JSR  $EB63
```
FAC vers ARG.

```
308 :  A0 02       LDY  £$02
30A :  20 01 E3    JSR  $E301
```
Y flottant dans FAC.

```
30D :  20 82 E9    JSR  $E982
```
FAC multiplié par ARG.

```
310 :  20 52 E7    JSR  $E752
```
On a le nombre sur 2 octets dans Y et A.

```
313 :  98          TYA
```
La partie basse passe dans A.

```
314 :  09 B0       ORA  £$B0
```
Bit 7 à 1.

```
316 :  20 ED FD    JSR  $FDED
```
Affichage.

```
319 :  2C 10 C0    BIT  $C010
31C :  AD 00 C0    LDA  $C000
31F :  2C 10 C0    BIT  $C010
322 :  C9 1B       CMP  £$1B
324 :  D0 DA       BNE  $0300
```
On continue aussi longtemps que la touche ESCAPE n'a pas été pressée.

```
326 :  60          RTS
```
Retour.

VOIR DÉMO SUR VOTRE DISQUETTE.

ALLER-RETOUR

Vous communiquez directement — ou presque ! — avec les routines arithmétiques du 65CØ2, mais comment récupérer le nombre obtenu dans une variable ? Tout bêtement en fournissant au microprocesseur le nom de ladite variable — ici N3. Quand le calcul est effectué $DFE3 (PTRGET) va rechercher cette variable et éventuellement la créer. Sa valeur sera pointée par VARPNT ($83-84). MOVMF ($EB2D) se chargera de transférer la valeur de FAC à cette adresse... si vous prenez la précaution — ô combien élémentaire — de mettre la partie basse de l'adresse en question dans X et sa partie haute dans Y.

```
100  TEXT : NORMAL :D$ = CHR$ (4): PRINT D$"PR£3": PRINT
110  FOR I = 768 TO 976: READ R: POKE I,R: IF R< > Ø THEN NEXT
120  HOME : PRINT "A L L E R - R E T O U R ":
          PRINT "_____"
130  PRINT : VTAB 12: CALL - 958
140  VTAB 14: INPUT "N1—» ";R$:N1 = VAL (R$): IF NOT N1 THEN 190
150  VTAB 16: INPUT "N2—» ";R$:N2 = VAL (R$): IF NOT N2 THEN 140
160  CALL 768,N1,N2,N3
170  PRINT N3
180  CALL - 198 : POKE 49168,Ø: WAIT 49152,128: POKE 49168,Ø: IF PEEK (49152) < > 27
     THEN 130
190  VTAB 22: PRINT : PRINT "(E)NCORE  (M)ENU DE DISQUETTE  (A)PPLESOFT ";: GET R$
200  IF R$ = "E" OR R$ = "e" THEN 130
210  IF R$ = "M" OR R$ = "m" THEN PRINT D$"RUN MENU"
220  IF R$ = "A" OR R$ = "a" THEN HOME : END
230  GOTO 190
240  DATA 32,190,222,32,123,221,32,99,235,32,190,222,32,123,221,32,130,233,32,190,222,
     32,227,223,166,131,164,132,76,45,235,Ø
```

```
300 :   20  BE  DE     JSR  $DEBE   ┐ N1 dans FAC.
303 :   20  7B  DD     JSR  $DD7B   ┘
306 :   20  63  EB     JSR  $EB63     FAC dans ARG.
309 :   20  BE  DE     JSR  $DEBE   ┐ N2 dans FAC.
30C :   20  7B  DD     JSR  $DD7B   ┘
30F :   20  82  E9     JSR  $E982     On divise, mais on pourrait faire n'importe quoi...
312 :   20  BE  DE     JSR  $DEBE   ┐ Adresse de la valeur de N3 dans VARPNT.
315 :   20  E3  DF     JSR  $DFE3   ┘
318 :   A6  83         LDX  $83     ┐ VARPNT dans X et Y.
31A :   A4  84         LDY  $84     ┘
31C :   4C  2D  EB     JMP  $EB2D     MOVMF transfère la valeur.  ■
```

GETNUM

Ce GET n'autorise que les chiffres de Ø à 9, à l'exclusion de toute autre touche.

```
100  TEXT :D$ = CHR$ (4): HOME :A = 767
110  A = A + 1: READ R: POKE A,R: IF A < 811 THEN 110
120  VTAB 9: PRINT "NUMERO entre Ø et 9 (Ø pour terminer)
130  PRINT : VTAB 12: CALL - 958
140  CALL 768,R
150  PRINT R
160  IF PEEK (37) = 20 THEN 130
170  IF R <> Ø THEN 140
180  VTAB 23: PRINT "(1) ENCORE  (2) MENU  (3) APPLESOFT"
190  CALL 768,R
200  IF R = 1 THEN VTAB 11: GOTO 130
210  IF R = 2 THEN PRINT D$"RUN MENU"
220  IF R <> 3 THEN 190
230  HOME
240  DATA 32,58,255,44,16,192,32,168,252,173,Ø,192,16,248,44,16,192,201,
     176,144,241,201,186,176,237,41,15,168,32,1,227,32,190,222,32,227,
     223,166,131,164,132,76,45,235
```

■

```
300 :  20 3A FF    JSR  $FF3A    BELL : émet son bip habituel.
303 :  2C 10 C0    BIT  $C010    KBDSTRB : échantillonnage du clavier.
306 :  20 A8 FC    JSR  $FCA8    WAIT : boucle d'attente selon valeur de A.
309 :  AD 00 C0    LDA  $C000    KBD : contient le code du caractère tapé (bit à 1).
30C :  10 F8       BPL  $0306    Pas de touche pressée : on continue d'attendre.
30E :  2C 10 C0    BIT  $C010    Pour un nouvel échantillonnage.
311 :  C9 B0       CMP  £$B0     ]
313 :  90 F1       BCC  $0306    ] Si plus petit que $B0 (0), refusé.
315 :  C9 BA       CMP  £$BA     ]
317 :  B0 ED       BCS  $0306    ] Si plus grand ou égal $BA (:), refusé itou.
319 :  29 0F       AND  £$0F     Transformation en 0,1,2, etc.
31B :  A8          TAY           A dans Y.
31C :  20 01 E3    JSR  $E301    L'entier de Y devient flottant.
31F :  20 BE DE    JSR  $DEBE    CHKCOM : teste la virgule en TXTPTR.
322 :  20 E3 DF    JSR  $DFE3    PTRGET : recherche la variable R (Call 768,R) ou la crée.
325 :  A6 83       LDX  $83      ]
327 :  A4 84       LDY  $84      ] L'adresse de sa valeur est en VARPNT.
329 :  4C 2D EB    JMP  $EB2D    MOVMF : transfère FAC vers l'adresse pointée par X,Y.
                                 R contient alors le chiffre tapé.
```

■

MIDNUM

Simple exemple de manipulation de variables. Vous tapez une série de nombres, mais la routine ne vous renvoie que le second. **Attention** ! aucun contrôle n'étant effectué, si vous tapez **1E3**, vous aurez **5** en retour. Pourquoi ? Parce que **E**, dans la machine, c'est $45 .

```
100 TEXT :D$ = CHR$ (4): HOME :A = 767
110 A = A + 1: READ R: POKE A,R: IF R <> 0 THEN 110
120 VTAB 9: PRINT "NOMBRE de plusieurs chiffres (le second seul sera pris
    en compte)"
130 PRINT : VTAB 12: CALL - 958: INPUT "—» ";R$: IF LEN (R$) < 2 OR NOT
    VAL (R$) THEN 130
140 CALL 768,R$,R
150 PRINT R
160 IF R = 0 THEN 180
170 CALL - 198: POKE 49168,0: WAIT 49152,128: POKE 49168,0: IF PEEK
    (49152) <> 27 THEN 130
180 VTAB 23: PRINT "(1) ENCORE  (2) MENU  (3) APPLESOFT ";: GET R$:R
    = VAL (R$)
190 IF R = 1 THEN VTAB 11: GOTO 130
200 IF R = 2 THEN PRINT D$"RUN MENU"
210 IF R <> 3 THEN 180
220 HOME
230 DATA  32,190,222,32,227,223,160,2,177,131,133,7,136,177,131,133,6,177,
    6,41,15,168,32,1,227,32,190,222,32,227,223,166,131,164,132,76,45,235
```

```
300 :  20 BE DE    JSR  $DEBE   ┐ On saute la virgule et on recherche la
303 :  20 E3 DF    JSR  $DFE3   ┘ variable R$.
306 :  A0 02       LDY  £$02    ┐
308 :  B1 83       LDA  ($83),Y │
30A :  85 07       STA  $07     │ L'adresse où est stockée la chaîne R$ est lue
30C :  88          DEY          │ en $83-84 + 1 et + 2. On la sauve en $6-7.
30D :  B1 83       LDA  ($83),Y │
30F :  85 06       STA  $06     ┘
311 :  B1 06       LDA  ($06),Y  Lecture du second caractère (Y = 1).
313 :  29 0F       AND  £$0F     On ne conserve que la partie faible $31 = $1.
315 :  A8          TAY           Puis on la passe dans Y.
316 :  20 01 E3    JSR  $E301    Il est flottant dans FAC.
319 :  20 BE DE    JSR  $DEBE   ┐ Repérage de R.
31C :  20 E3 DF    JSR  $DFE3   ┘
31F :  A6 83       LDX  $83     ┐ L'adresse de R est placée dans X et Y.
321 :  A4 84       LDY  $84     ┘
323 :  4C 2D EB    JMP  $EB2D    FAC est transféré dans R. Terminé ! ■
```

MIN.BAS

Si vous avez un jour besoin d'éliminer la ponctuation, les accents circonflexes et les trémas d'un texte... tout en transformant les éventuelles lettres minuscules et accentuées en CAPITALES, utilisez MINCAPBUF.1.

```
100 TEXT :D$ = CHR$ (4): PRINT D$"BLOAD MIN/MINCAPBUF.1": PRINT
    D$"PR£3": PRINT
110 PRINT : PRINT "Saisie en minuscules ou capitales, avec la ponctuation,
    mais le texte utilisé sera présenté en capitales et sans ponctuation."
120 PRINT : VTAB 8: CALL - 958: PRINT "TEXTE: ";: CALL - 657
130 GOSUB 230: IF A$ = "" THEN HOME : GOTO 170
140 PRINT A$: CALL 768: GOSUB 230
150 PRINT A$
160 :
170 VTAB 22: PRINT "(E)NCORE (M)ENU DE DISQUETTE (A)PPLESOFT ";:
    CALL - 657: GOSUB 230
180 CALL 776: GOSUB 230
190 IF A$ = "E" THEN 120
200 IF A$ = "M" THEN PRINT D$"RUN MENU,D1
210 IF A$ = "A" THEN HOME : END
220 GOTO 170
230 A$ = "": FOR X = 512 TO 767: IF PEEK (X) < > 141 THEN A$ = A$ +
    CHR$ ( PEEK (X) - 128): NEXT
240 RETURN
```

MINCAPBUF.1

Accentuées ou non, les minuscules deviennent des capitales. L'accent circonflexe et le tréma sont éliminés, ainsi que la ponctuation.

```
300 :   A2 00       LDX  £$00      REGISTRE X à 0.
302 :   BD 00 02    LDA  $0200,X   Lecture du caractère dans $200 + X.
305 :   C9 DE       CMP  £$DE  ]
307 :   F0 37       BEQ  $0340 ]   Accent circonflexe ? oui : traitement spécial.
309 :   C9 FE       CMP  £$FE  ]
30B :   F0 33       BEQ  $0340 ]   Tréma : même traitement.
30D :   C9 C0       CMP  £$C0      Test du à.
30F :   D0 02       BNE  $0313 ]
311 :   A9 C1       LDA  £$C1  ]   Saut ou à devient A.
```

```
313 :  C9 FC      CMP  £$FC      Test du ù.
315 :  DØ Ø2      BNE  $Ø319     ┐ Saut ou ù devient U.
317 :  A9 D5      LDA  £$D5      ┘
319 :  C9 FB      CMP  £$FB      Test du é.
31B :  FØ Ø4      BEQ  $Ø321     ┐ Si oui, saut.
31D :  C9 FD      CMP  £$FD      ┘ Test du è.
31F :  DØ Ø2      BNE  $Ø323     ┐ Saut ou bien é-è deviennent E.
321 :  A9 C5      LDA  £$C5      ┘
323 :  C9 DC      CMP  £$DC      Test du ç.
325 :  DØ Ø2      BNE  $Ø329     ┐ Saut ou bien ç devient C.
327 :  A9 C3      LDA  £$C3      ┘
329 :  C9 8D      CMP  £$8D      ┐ Si c'est RETURN, terminé.
32B :  FØ 12      BEQ  $Ø33F     ┘
32D :  29 DF      AND  £$DF      $DF = 11011110 force des bits à Ø.
32F :  C9 C1      CMP  £$C1      ┐ Plus petits que A($C1), refusé.
331 :  9Ø Ø4      BCC  $Ø337     ┘
333 :  C9 DB      CMP  £$DB      ┐ Plus petits que $DB(Z+1), c'est bon :
335 :  9Ø Ø2      BCC  $Ø339     ┘ saut direct.
337 :  A9 AØ      LDA  £$AØ      Espace remplace les caractères rejetés.
339 :  9D ØØ Ø2   STA  $Ø2ØØ,X   Transformation dans le buffer.
33C :  E8         INX            ┐ On continue en tout cas.
33D :  8Ø C3      BRA  $Ø3Ø2     ┘
33F :  6Ø         RTS            Retour au Basic.
```

FLEXE ET TRÉMA

```
34Ø :  8A         TXA            ┐ Avec le 65C816, TYX ($9B) fait la même
341 :  A8         TAY            ┘ chose.
342 :  C8         INY            Y = X + 1, si l'on préfère.
343 :  B9 ØØ Ø2   LDA  $Ø2ØØ,Y   ┐ Il s'agit de remplacer l'accent par le sui-
346 :  9D ØØ Ø2   STA  $Ø2ØØ,X   ┘ vant, etc.
349 :  C9 8D      CMP  £$8D      ┐ RETURN, retour à la case départ.
34B :  FØ B3      BEQ  $Ø3ØØ     ┘
34D :  E8         INX            ┐ Vers Y + 1.
34E :  DØ F2      BNE  $Ø342     ┘
35Ø :  8Ø AE      BRA  $Ø3ØØ     FIN.
```

■

BANDE PASSANTE

Utilisation de la routine $FC7Ø (SCROLL).

Taper **BRUN BANDE.PASSANTE** ou CALL 768 pour obtenir l'effacement de l'écran.

```
0300-  A9 17      LDA  £$17
0302-  AA         TAX
0303-  20 47 F8   JSR  $F847
0306-  A0 28      LDY  £$28
0308-  A9 AA      LDA  £$AA
030A-  91 26      STA  ($26),Y
030C-  88         DEY
030D-  10 FB      BPL  $030A
030F-  20 70 FC   JSR  $FC70
0312-  CA         DEX
0313-  10 FA      BPL  $030F
0315-  60         RTS
```

MINCAPBUF.0

Routine simplifiée sans traitement des accents circonflexes et des trémas.

```
300 :  A2 00        LDX  £$00      Registre X = 0.
302 :  BD 00 02     LDA  $0200,X   Caractère trouvé à l'adresse $200 + X mis
                                   dans l'accumulateur.
305 :  C9 C0        CMP  £$C0      ┐ Si ce n'est pas le à, saut.
307 :  D0 02        BNE  $030B     ┘
309 :  A9 C1        LDA  £$C1      Le à devient A (INC A - code 1A - fait la
                                   même chose sur le 65C02).
30B :  C9 FC        CMP  £$FC      Est-ce ù ?
30D :  D0 02        BNE  $0311     Non : on va voir plus loin.
30F :  A9 D5        LDA  £$D5      Oui : ù devient U.
311 :  C9 FB        CMP  £$FB      Est-ce é ?
313 :  D0 02        BNE  $0317     Non : voir la suite.
315 :  A9 C5        LDA  £$C5      Oui : é devient E.
317 :  C9 FD        CMP  £$FD      Est-ce è ?
319 :  D0 02        BNE  $031D     Toujours le saut si la réponse est négative.
31B :  A9 C5        LDA  £$C5      Oui : è devient E.
31D :  C9 DC        CMP  £$DC      Est-ce un ç ?
31F :  D0 02        BNE  $0323     Saut ou pas saut.
321 :  A9 C3        LDA  £$C3      Oui : ç devient C.
323 :  C9 8D        CMP  £$8D      Est-ce RETURN ?
325 :  F0 12        BEQ  $0339     Si oui on a terminé.
327 :  29 DF        AND  £$DF      DF = 11011110 force les bits à 0.
329 :  C9 C1        CMP  £$C1      ┐ Si le caractère est plus petit que $C1, saut
32B :  90 04        BCC  $0331     ┘ vers $A0.
32D :  C9 DB        CMP  £$DB      ┐ Si le caractère est plus petit que $DB, il est
32F :  90 02        BCC  $0333     ┘ bon (de A à Z).
331 :  A9 A0        LDA  £$A0      ┐ Sinon on le transforme en espace et on le
333 :  9D 00 02     STA  $0200,X   ┘ met à la place de l'ancien.
336 :  E8           INX            ┐ On continue...
337 :  80 C9        BPL  $0302     ┘
339 :  60           RTS            Retour au Basic.                    ■
```

- On peut aussi utiliser une table permettant une permutation facile des caractères à transcoder. Lisez l'exemple page suivante.

MINCAPBUF.2

Utilisation d'une table de transcodage.

```
300 :  A2 00        LDX  £$00      ┐ Registre X = 0.
302 :  BD 00 02     LDA  $0200,X   ┘ Caractère trouvé à l'adresse $200 + X mis
                                     dans l'accumulateur.
```

COMPARAISON DU CARACTÈRE AVEC LA TABLE

```
305 :  A0 08        LDY  £$08      Registre Y=8 (on va avoir Y=8, 6, 4, 2, 0).
307 :  D9 2C 03     CMP  $032C,Y   Comparaison avec le caractère de la table.
30A :  D0 05        BNE  $0311     Ce n'est pas le même : saut.
30C :  B9 2D 03     LDA  $032D,Y   On prend le caractère qui remplace
                                   (adresse + 1).
30F :  D0 14        BNE  $0325     Saut à Permutation.
311 :  88           DEY            ┐
312 :  88           DEY            ┘ Y = Y - 2.
313 :  10 F2        BPL  $0307     Suite jusqu'à Y=0 inclus.
```

EST-CE RETURN ?

```
315 :  C9 8D        CMP  £$8D      Est-ce RETURN ?
317 :  F0 12        BEQ  $032B     Si oui on a terminé.
```

SUITE

```
319 :  29 DF        AND  £$DF      DF = 11011110 force les bits à 0.
31B :  C9 C1        CMP  £$C1      ┐ Si le caractère est plus petit que $C1, saut
31D :  90 04        BCC  $0323     ┘ vers $A0.
31F :  C9 DB        CMP  £$DB      ┐ Si le caractère est plus petit que $DB, il est
321 :  90 02        BCC  $0325     ┘ bon (de A à Z).
323 :  A9 A0        LDA  £$A0      Sinon on le transforme en espace.
```

PERMUTATION

```
325 :  9D 00 02     STA  $0200,X
328 :  E8           INX            ┐
329 :  80 D7        BRA  $0302     ┘ On continue...
32B :  60           RTS            Retour au Basic.
```

TABLE DE TRANSCODAGE

```
32C :  C0 C1   à / A
       FC D5   ù / U
       FB C5   é / E
       FD C5   è / E
       DC C3   ç / C
```

■

PROBLÈME

Je désire afficher un carré de points dont le côté varierait de 1 à 23, mais ***sans passer par le Basic.*** J'aimerais aussi pouvoir récupérer le nombre de points, en hexadécimal, dans la page Ø. J'avoue que je souhaite une solution facile à comprendre, même si elle doit se traduire par un programme assez long.

VOICI EFFECTIVEMENT UNE SOLUTION...

```
2F5 :  C4 C9
2F7 :  CD C5 CE
2FA :  D3 C9
2FC :  CF CE BA A0
```

DIMENSION : mot + 1 espace.

Le texte est placé dans le buffer... où il ne gêne pas, mais pourrait aussi bien prendre place à partir de l'adresse $378.

DÉBUT DU PROGRAMME

```
300 :  A9 15      LDA £15
302 :  20 ED FD   JSR FDED
305 :  20 58 FC   JSR FC58
```

PRINT CHR$ (21) pour passer en 40 colonnes.

HOME.

TEXTE (CÔTÉ DU CARRÉ)

```
308 :  B9 F5 02   LDA 02F5,Y
30B :  20 ED FD   JSR FDED
30E :  C8         INY
30F :  C0 0B      CPY £0B
311 :  D0 F5      BNE 0308
313 :  A9 DF      LDA £DF
315 :  20 ED FD   JSR FDED
318 :  20 10 FC   JSR FC10
```

Affichage du texte.

Souligné pour le curseur (fantaisie inutile).

On recule d'une case (pour avoir le souligné sous le prompt).

SAISIE DE LA VALEUR

```
31B :  20 6F FD   JSR FD6F
```

A la sortie, X = nombre de caractères (si on a plus de 2 chiffres seuls comptent les 2 premiers).

```
31E :  AD 01 02   LDA 0201
321 :  29 0F      AND £0F
323 :  85 07      STA 07
325 :  AD 00 02   LDA 0200
328 :  29 0F      AND £0F
32A :  CA         DEX
32B :  F0 0A      BEQ 0337
```

Chiffre de poids faible stocké en $7
avec AND 00001111, $B1 devient 1, etc.

Chiffre de poids fort (éventuellement unique).

Si X = Ø, on se passe de tout calcul.

IL NOUS FAUT DE L'HEXA

```
32D :  0A         ASL
32E :  85 06      STA 06
330 :  0A         ASL
331 :  0A         ASL
332 :  18         CLC
333 :  65 07      ADC 07
335 :  65 06      ADC 06
```

Multiplie A par deux.
Mémo provisoire.

Multiplie A par 4.

Retenue pour additionner.

Double addition.

CONTRÔLE

```
337 : C9 18     CMP £18
339 : B0 C5     BCS 0300
33B : C9 00     CMP £00
33D : F0 C1     BEQ 0300
33F : 85 06     STA 06
```

Si le nombre binaire obtenu est supérieur à $17 ou égal à 0, à refaire.

Côté du carré (en hexa).

SURFACE

```
341 : 20 93 EB  JSR EB93
344 : 20 63 EB  JSR EB63
347 : A5 06     LDA 06
349 : 20 93 EB  JSR EB93
34C : 20 82 E9  JSR E982
34F : 20 52 E7  JSR E752
352 : 84 07     STY 07
354 : 85 08     STA 08
```

Contenu de A flottant dans FAC.
FAC dans ARG.
On reprend le même...
... pour multiplier FAC par ARG.
Transformation sur 2 octets (nombre de cases).
On lira éventuellement ce nombre de points en $7/8.

AFFICHAGE

```
356 : A6 06     LDX 06
358 : A4 06     LDY 06
35A : A9 AE     LDA £AE
35C : 20 ED FD  JSR FDED
35F : 88        DEY
360 : D0 FA     BNE 035C
362 : 20 62 FC  JSR FC62
365 : CA        DEX
366 : D0 F0     BNE 0358
```

Côté du carré dans X et Y.
Le point (.) dans A.
Affichage.
Suite jusqu'à Y points.
On passe à la ligne suivante.
X = X - 1.
S'il est différent de 0, on continue.

PAUSE

```
368 : 2C 10 C0  BIT C010
36B : AD 00 C0  LDA C000
36E : 10 FB     BPL 036B
370 : 2C 10 C0  BIT C010
373 : C9 9B     CMP £9B
375 : D0 89     BNE 0300
377 : 60        RTS
```

Lecture du clavier.
Si ESCAPE, retour au Basic.

■

SLMO

Une autre manière de saisir un nombre à partir d'une routine LM. On le récupère en $6 (limite : 255). Un message d'erreur sanctionnera le dépassement, et ProDOS n'aimera pas cela !

```
300 : 20 6F FD  JSR $FD6F
303 : BD FF 01  LDA $01FF,X
306 : 29 3F     AND £$3F
308 : 9D 00 02  STA $0200,X
30B : CA        DEX
30C : D0 F5     BNE $0303
```

GETLIN sans son astérisque.

AND $3F 0011 1111 pour remplacer l'ASCII par des caractères adéquats, avec décalage d'un caractère vers la droite.

(suite page 76)

```
30E :  8A          TXA
30F :  85 B8       STA  $B8        x = 0.
311 :  A9 02       LDA  £$02       On trompe ainsi CHRGET.
313 :  85 B9       STA  $B9
315 :  20 F5 E6    JSR  $E6F5      GETBYTC saute un caractère et on va retrouver
318 :  86 06       STX  $06        la valeur saisie (en hexa) dans X.
31A :  4C B5 03    JMP  $03B5      Le JMP concerne la disquette (RUN MENU).
```

SLM1

Ici, on peut saisir des valeurs quelconques. Le nombre est multiplié par lui-même, comme dans **PROBLÈME**.

```
300 :  A9 BD       LDA  £$BD       Signe égal dans A, puis appel de GETLIN
302 :  20 6C FD    JSR  $FD6C      (astérisque remplacé par =).
305 :  8A          TXA
306 :  F0 40       BEQ  $0348      Si x = 0, terminé.
308 :  A9 00       LDA  £$00
30A :  9D 01 02    STA  $0201,X    On finit par un zéro dans le buffer.
30D :  BD FF 01    LDA  $01FF,X
310 :  29 3F       AND  £$3F
312 :  9D 00 02    STA  $0200,X    Voir SLM0.
315 :  CA          DEX
316 :  D0 F5       BNE  $030D
318 :  8A          TXA
319 :  85 B8       STA  $B8
31B :  A9 02       LDA  £$02       Il s'agit encore de tromper CHRGET.
31D :  85 B9       STA  $B9
31F :  A9 2C       LDA  £$2C       Si CHKCOM ne trouve pas de virgule, il ne
321 :  8D 00 02    STA  $0200      sera pas content.
324 :  20 BE DE    JSR  $DEBE      CHKCOM trouve bien sa virgule.
327 :  20 7B DD    JSR  $DD7B      FRMEVL évalue la formule.
32A :  20 63 EB    JSR  $EB63      MOVAF : FAC dans ARG.
32D :  20 21 EB    JSR  $EB21      MOV1F : TEMPS1/FAC.
330 :  20 82 E9    JSR  $E982      FMULT : ARG × FAC.
333 :  20 2E ED    JSR  $ED2E      PRNTFAC affiche la valeur de FAC (décimale).
336 :  20 62 FC    JSR  $FC62      CR (retour chariot).
339 :  2C 10 C0    BIT  $C010
33C :  AD 00 C0    LDA  $C000
33F :  10 FB       BPL  $033C      Attente et test d'ESCAPE
341 :  2C 10 C0    BIT  $C010      (qui permet de terminer). Voir PROBLÈME.
344 :  C9 9B       CMP  £$9B
346 :  D0 B8       BNE  $0300
348 :  4C B5 03    JMP  $03B5      RUN MENU.                              ■
```

SLM2

On retrouve le nombre dans une variable A (pas plus de 255).

```
5  PRINT CHR$ (4)"PREFIX/R4/PROBL": HOME
10 VTAB 12: PRINT "TAPEZ (BRUN SLM2), puis entrez un .....
                   nombre compris entre 1 et 255 .....
                   TAPEZ ensuite (PRINT A) ..................."
20 VTAB 18: PRINT "§BRUNSLM2": VTAB 20: PRINT "§PRINTA": VTAB 23:
   PRINT "§RUN MENU,S6": VTAB 17
```

** Les points de la ligne 10 représentent des espaces.*

SAISIE DE LA DONNÉE

```
300 :  A9 BE      LDA  £$BE
302 :  20 6C FD   JSR  $FD6C
```

Le signe ***plus grand que*** est affiché et GETLIN attend...

MODIFICATION DU BUFFER

```
305 :  A9 2C      LDA  £$2C
307 :  9D 01 02   STA  $0201,X
30A :  A9 41      LDA  £$41
30C :  9D 02 02   STA  $0202,X
30F :  A9 00      LDA  £$00
311 :  9D 03 02   STA  $0203,X
314 :  BD FF 01   LDA  $01FF,X
317 :  29 3F      AND  £$3F
319 :  9D 00 02   STA  $0200,X
31C :  CA         DEX
31D :  D0 F5      BNE  $0314
31F :  8A         TXA
320 :  85 B8      STA  $B8
322 :  A9 02      LDA  £$02
324 :  85 B9      STA  $B9
```

Virgule dans le buffer à $201 + nbre de caractères.

A (nom de la variable) à $202 + nbre de caractères.

0 pour terminer.

Chaque caractère est transformé par AND 0011 1111 qui met les bits 6 et 7 à 0, et décalé d'une case vers la droite.

X (qui vaut 0) dans A.

$200 pour tromper CHRGET.

ANALYSE

```
326 :  20 F5 E6   JSR  $E6F5
329 :  86 06      STX  $06
32B :  20 BE DE   JSR  $DEBE
32E :  20 E3 DF   JSR  $DFE3
```

GETBYTC puis GETBYT (premier caractère négligé) : résultat dans $06 via X.

CHKCOM teste la virgule et PTRGET recherche la variable ou la crée (ici, il la crée).

TRANSFERT DANS A

```
331 :  A4 06      LDY  $06
333 :  20 01 E3   JSR  $E301
336 :  A6 83      LDX  $83
338 :  A4 84      LDY  $84
33A :  20 2D EB   JSR  $EB2D
33D :  4C 6F F2   JMP  $F26F
```

Récupération de la valeur rentrée au clavier. SGNFLT rend cette valeur flottante.

Adresse de la variable A.

MOVMF transfère FAC dans l'adresse ci-dessus. NOTRACE pour retour. ■

DAT.AFF

Affichage de la date sous la forme JJ MM AA (en admettant que ProDOS la connaisse déjà... ou tourne avec une version récente et une carte.horloge).

UTILISATION : BRUN DAT.AFF

```
300 : A9 16     LDA £$16     $16 dans A pour VTAB.
302 : 20 5B FB  JSR $FB5B    Curseur en $16.
305 : 20 9C FC  JSR $FC9C    CALL - 868 du Basic (ligne effacée).
308 : 20 00 BF  JSR $BF00    Entrée.
30B : CC 3C 03               $CC = CLOSE — $33C = adresse paramètre.
30E : AD 90 BF  LDA $BF90  ┐
311 : 48        PHA        │
312 : 29 1F     AND £$1F   │
314 : 85 06     STA $06    │
316 : AD 91 BF  LDA $BF91  │
319 : 85 08     STA $08    │
31B : 46 08     LSR $08    │  Voir explications dans DAT.VAR.
31D : 68        PLA        │
31E : 6A        ROR        │
31F : 4A        LSR        │
320 : 4A        LSR        │
321 : 4A        LSR        │
322 : 4A        LSR        │
323 : 85 07     STA $07    ┘
325 : A2 00     LDX £$00     x = 0.
327 : B5 06     LDA $06,X    Lecture du JOUR, puis du mois et de l'année.
329 : DA        PHX          X empilé.
32A : 20 93 EB  JSR $EB93    L'entier qui est dans A devient flottant.
32D : 20 2E ED  JSR $ED2E    PRNTFAC affiche la valeur de FAC.
330 : A9 A0     LDA £$A0   ┐  Un espace.
332 : 20 ED FD  JSR $FDED  ┘
335 : FA        PLX          Récupération de X.
336 : E8        INX          X = X + 1.
337 : E0 03     CPX £$03   ┐  On continue jusqu'à X = 2 inclus.
339 : D0 EC     BNE $0327  ┘
33B : 60        RTS          FIN.
33C : 01 04                  1 seul paramètre — 4 = Fichiers texte. ■
```

DAT.VAR

La date (si ProDOS la connaît) revient dans une variable. Au départ, elle est dans les octets $BF90.BF91, sous la forme ci-dessous :

$BF91	$BF90	
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	JOUR : 0 à 4 de $BF90
A A A A A A A M	M M M J J J J J	MOIS : 5 à 7 de $BF90 + 0 de $BF91
		ANNÉE : 1 à 7 de $BF91

PROCÉDURE : PRINT CHR$ (4)"BLOAD DAT.VAR": CALL 768,A$: PRINT A$

```
300:  20 00 BF   JSR  $BF00     Point d'entrée.
303:  CC         $CC  ]
304:  5E 03      $35E ]         $CC = close -- $35E : adresse des paramètres.
306:  AD 90 BF   LDA  $BF90     Lecture SYSTEM TIME.
309:  48         PHA            Valeur empilée pour plus tard.
30A:  29 1F      AND  £$1F  ]   AND 0001 1111 : on conserve les bits 0 à 4 pour
30C:  85 06      STA  $06   ]   le JOUR. Stockage en $6.
30E:  AD 91 BF   LDA  $BF91 ]   Lecture second octet.
311:  85 08      STA  $08   ]   Stockage en $08.
313:  46 08      LSR  $08       On décale (un zéro entre à gauche et le bit sortant
                                tombe dans la retenue : on a l'ANNÉE en $8).
315:  68         PLA            Récupération sur la pile.
316:  6A         ROR            Le bit de retenue entre à gauche.
317:  4A         LSR  ]
318:  4A         LSR  ]         Encore quatre bits à éliminer
319:  4A         LSR  ]         (des zéros entrent à gauche).
31A:  4A         LSR  ]
31B:  85 07      STA  $07       Mois en 7.
31D:  A2 00      LDX  £$00  ]   Préparation.
31F:  A0 FF      LDY  £$FF  ]
321:  84 09      STY  $09       $9 contiendra la valeur de Y.
323:  B4 06      LDY  $06,X     Lecture du JOUR (hexa), puis du MOIS... et de
                                l'ANNÉE.
325:  DA         PHX            X entassé sur la pile.
326:  20 01 E3   JSR  $E301     SGNFLT rend flottant l'entier de Y (dans FAC).
329:  20 34 ED   JSR  $ED34     FOUT convertit FAC en chaîne de chiffres.
32C:  A2 00      LDX  £$00      X = 0.
32E:  A4 09      LDY  $09       Y récupéré dans $9.
```

(suite page 80)

```
330 :  C8          INY          Y = Y + 1.
331 :  BD 00 01    LDA $0100,X  On lit ici la chaîne de chiffres.
334 :  F0 06       BEQ $033C    Si c'est 0, c'est fini.
336 :  99 C0 03    STA $03C0,Y  Adresse de stockage.
339 :  E8          INX          ] On continue obligatoirement.
33A :  D0 F4       BNE $0330    ]
33C :  A9 20       LDA £$20     ] Un blanc pour séparer le JOUR, du mois, de
33E :  99 C0 03    STA $03C0,Y  ] l'année.
341 :  FA          PLX          X est récupéré sur la pile.
342 :  E8          INX          X = X + 1.
343 :  E0 03       CPX £$03     ] On continue jusqu'à X = 2 inclus.
345 :  D0 DA       BNE $0321    ]
347 :  20 BE DE    JSR $DEBE    ] Teste de la virgule et recherche de la variable.
34A :  20 E3 DF    JSR $DFE3    ]
34D :  A0 00       LDY £$00     Y = 0.
34F :  A9 08       LDA £$08     ] Longueur de la date dant le descripteur.
351 :  91 83       STA ($83),Y  ]
353 :  C8          INY          Y = 1.
354 :  A9 C0       LDA £$C0     ] Partie basse de l'adresse.
356 :  91 83       STA ($83),Y  ]
358 :  C8          INY          Y = 2.
359 :  A9 03       LDA £$03     ] Partie haute de l'adresse.
35B :  91 83       STA ($83),Y  ]
35D :  60          RTS          Retour.
35E :  01 04                    Paramètre : type de fichier = 4.
```

TRI.B

Comment trier rapidement une liste de nombres sur un octet ? Jusqu'à 255, voici une routine classique*, pouvant d'ailleurs servir de base à d'autres, plus complexes. Elle sous-entend que la liste à trier est installée à partir de l'adresse $2000... et que le nombre d'éléments est contenu dans la case-mémoire $1FFF. Dans notre démonstration, les nombres sont aléatoirement mis en place par la ligne Basic 20. Nous nous sommes limités, ici, à 95 pour simplifier l'affichage, mais vous pouvez essayer avec d'autres valeurs (ne dépassez pas 255). La vérification ($302 à $308) pourrait être escamotée dans un programme où l'on aurait la certitude d'avoir plus d'un élément. Il n'est pas davantage interdit de placer le nombre d'éléments moins un (NE - 1) dans $1FFF, ce qui élimine le DEY de la ligne $30C.

** Elle n'est certainement pas inédite, soit dans cette version, soit dans des versions voisines.*

```
10  TEXT : NORMAL :D$ = CHR$ (4):
    PRINT D$"PR£3": PRINT : PRINT
    D$"BLOAD TRI/TRI": HOME

20  FOR I = 8192 TO 8192 + 94: POKE I,1
    + INT ( RND (1) * 95): NEXT : POKE
    8191,95

30  A$ = "PRESSEZ UNE TOUCHE POUR
    ACTIVER LE TRI ": GOSUB 100

40  CALL 768 : HOME

50  A$ = "VOTRE LISTE EST TRIEE (PAR
    COLONNE) ": GOSUB 100

60  A$ = "(T)RI POUR UNE AUTRE
    LISTE ": GOSUB 110

70  VTAB 23: PRINT "(M)ENU DE DIS
    QUETTE (A)PPLESOFT ";: GET R$:
    VTAB 22: PRINT

80  IF R$ = "M" OR R$ "m" THEN PRINT
    D$ "RUN MENU,S6"

90  IF R$ = "A" OR R$ = "a" THEN HOME
    : END

95  GOTO 60

100 FOR I = 8192 TO 8192 + 18: FOR J =
    I TO I + 76 STEP 19: PRINT PEEK (J),:
    NEXT : NEXT

110 VTAB 22: CALL - 958: PRINT : PRINT
    A$;: GET R$: VTAB 22: PRINT

120 IF R$ = "T" OR R$ = "t" THEN HOME :
    GOTO 20

130 RETURN
```

```
300 :  A2 00        LDX  £$00      Registre X à 0.

                    PRÉCAUTION

302 :  AC FF 1F     LDY  $1FFF     Nombre d'éléments à trier dans Y.
305 :  C0 02        CPY  £$02  ]   S'il n'y a pas de retenue, on n'a qu'un élément
307 :  90 1F        BCC  $0328 ]   et le tri est inutile.

                    TRI

309 :  AC FF 1F     LDY  $1FFF ]   Nombre d'éléments moins un dans Y (NE - 1).
30C :  88           DEY        ]
30D :  B9 00 20     LDA  $2000,Y ] Charger NE + Y dans l'Accumulateur
310 :  D9 FF 1F     CMP  $1FFF,Y ] et le comparer à NE - 2.
313 :  B0 0D        BCS  $0322     S'il n'y a pas de retenue, il est plus petit. S'il est
                                   plus grand, on saute.
315 :  A2 01        LDX  £$01      Puisqu'il y a échange, X est mis à 1.
317 :  48           PHA            A empilé (nombre à permuter).
318 :  B9 FF 1F     LDA  $1FFF,Y ] Le plus grand prend la place du plus petit.
31B :  99 00 20     STA  $2000,Y ]
31E :  68           PLA        ]   On retrouve celui-ci sur la pile et il prend la
31F :  99 FF 1F     STA  $1FFF,Y ] place du plus grand.
322 :  88           DEY        ]   Y = Y - 1. On continue jusqu'à Y = 1 inclus.
323 :  D0 E8        BNE  $030D ]
325 :  CA           DEX        ]   X = X - 1. Si X = 0, il y a eu permutation :
326 :  F0 E1        BEQ  $0309 ]   encore un tour !
328 :  60           RTS                                                    ■
```

UN x DEUX

Routine uniquement destinée à vous montrer comment il est possible de multiplier A par B... pour retrouver le résultat dans C, étant entendu que A B C peuvent être du type A(Ø) A(1) A(2) aussi bien que A(4) A(5) A(6)...

```
100 TEXT : NORMAL : HOME : GOSUB
    200
110 VTAB 9: PRINT "RENTREZ DEUX
    CHIFFRES ": PRINT
120 INPUT "A(Ø) —» ";A(Ø)
130 INPUT "A(1) —» ";A(1) : PRINT
140 CALL 768,A(Ø): PRINT : LIST 150
150 PRINT : PRINT A(2)
160 VTAB 22: PRINT "(E)ncore (M)enu
    (F)in ";: GET R$
170 IF R$ = "E" OR R$ = "e" THEN
    HOME : GOTO 110
175 IF R$ = "M" OR R$ = "m" THEN
    PRINT CHR$ (4)"RUN MENU"
180 IF R$ = "F" OR R$ = "f" THEN
    HOME : END
190 GOTO 160
200 FOR I = 768 TO 828: READ R:
    POKE I,R: NEXT : RETURN
210 DATA 32,190,222,165,184,72,165,
    185,72,32,227,223,133,6,132,7,
    104,133,185,104,133,184,32,103,
    221,32,99,235,164,7,165,6,24,
    105,5,144,1,200,32,249,234,32,
    130,233,164,7,165,6,24,105,10,
    170,144,1,200,32,45,235,76,46,237
```

```
300 :  20 BE DE   JSR $DEBE
303 :  A5 B8      LDA $B8
305 :  48         PHA
306 :  A5 B9      LDA $B9
308 :  48         PHA
309 :  20 E3 DF   JSR $DFE3
30C :  85 06      STA $06
30E :  84 07      STY $07
310 :  68         PLA
311 :  85 B9      STA $B9
313 :  68         PLA
314 :  85 B8      STA $B8
316 :  20 67 DD   JSR $DD67
319 :  20 63 EB   JSR $EB63
31C :  A4 07      LDY $07
31E :  A5 06      LDA $06
320 :  18         CLC
321 :  69 05      ADC £$05
```

Teste la présence de la virgule.

Position de TXTPTR (adresse du dernier caractère obtenu par CHRGET) sauvegardée sur la pile.

PTRGET recherche la variable (ici A(Ø)).

Son adresse est en A et Y. On la sauve en $6.7.

Restauration de CHRGET.

FRNUM évalue la formule pointée par TXTPTR et la met dans FAC.

FAC dans ARG.

Partie haute de l'adresse de a(Ø) dans Y.

Partie basse + 5 dans A pour adresse de A(1).

```
323 : 90 01       BCC $0326    ┐ S'il y a retenue, la partie haute (Y) est
325 : C8          INY          ┘ incrémentée.
326 : 20 F9 EA    JSR $EAF9      MOVFM place le nombre pointé par Y,A dans FAC.
329 : 20 82 E9    JSR $E982      FMULTT multiplie FAC par ARG.
32C : A4 07       LDY $07        Adresse haute dans Y.
32E : A5 06       LDA $06      ┐
330 : 18          CLC          │ Adresse basse majorée de $A (deux fois 5)
331 : 69 0A       ADC £$0A     │ dans X via A, pour adresse de A(2).
333 : AA          TAX          ┘
334 : 90 01       BCC $0337    ┐ Si retenue, on incrémente la partie haute.
336 : C8          INY          ┘
337 : 20 2D EB    JSR $EB2D      MOVMF transfère FAC vers le groupe de 5
                                 octets pointé par X et Y.
33A : 4C 2E ED    JMP $ED2E      Affiche la valeur de FAC (résultat de la
                                 multiplication).
```

ECRLETTRES/LO

Des lettres en mode inverse ou normal viennent progressivement remplir l'écran 40 colonnes. ESCAPE permet d'arrêter les frais.

```
300 : 20 58 FC    JSR $FC58      HOME efface l'écran.
303 : A9 18       LDA £$18     ┐ VTAB dans $6.
305 : 85 06       STA $06      ┘
307 : 20 4D 03    JSR $034D      Vers RND pour VTAB aléatoire.
30A : 98          TYA            VTAB dans A.
30B : 20 47 F8    JSR $F847      GBASCALC calcule l'adresse de base de la ligne A.
30E : A9 28       LDA £$28     ┐ HTAB dans $6.
310 : 85 06       STA $06      ┘
312 : 20 4D 03    JSR $034D      Vers RND pour HTAB aléatoire.
315 : 98          TYA            Y dans A.
316 : 18          CLC          ┐
317 : 65 26       ADC $26      │ Adresse de base + A pour HTAB aléatoire.
319 : 85 26       STA $26      ┘
31B : 90 02       BCC $031F    ┐ On incrémente la partie haute s'il y a retenue.
31D : E6 27       INC $27      ┘
31F : A9 1A       LDA £$1A     ┐ Pour caractère dans $6.
321 : 85 06       STA $06      ┘
```

```
323 : 20 4D 03    JSR  $034D     Vers RND pour carac.
326 : C8          INY            Y = Y + 1.
327 : 98          TYA            Y dans A.
328 : 48          PHA            On empile.
329 : A0 00       LDY  £$00      Y = 0.
32B : B1 26       LDA  ($26),Y   Que contient cette mémoire-écran ?
32D : C9 80       CMP  £$80      ]
32F : B0 04       BCS  $0335     ] Si c'est plus grand ou égal $80, voir plus loin...
331 : 68          PLA            Sinon, récupération de A.
332 : 09 C0       ORA  £$C0      Bits 7 et 6 forcés à 1.
334 : 48          PHA            On entasse.
335 : 68          PLA            Pour le retrouver ici
                                 (mais on peut arriver de $32F).
336 : 91 26       STA  ($26),Y   Caractère affiché (poke).
338 : A8          TAY            A dans Y.
339 : 2C 30 C0    BIT  $C030     CLICK SPEAKER.
33C : 88          DEY            Y décrémenté.
33D : 10 FA       BPL  $0339     On continue le bruit jusqu'à Y = 0 inclus.
33F : 2C 10 C0    BIT  $C010     Clavier à 0.
342 : AD 00 C0    LDA  $C000     Lecture clavier.
345 : 2C 10 C0    BIT  $C010     Remise à 0.
348 : C9 1B       CMP  £$1B      ]
34A : D0 B7       BNE  $0303     ] Si on a pressé ESCAPE, c'est fini ;
                                 sinon on continue.
34C : 60          RTS
```

RND

```
34D : A9 01       LDA  £$01      ]
34F : 20 B1 EF    JSR  $EFB1     ] Nombre aléatoire.
352 : 20 63 EB    JSR  $EB63     MOVAF transfère FAC dans ARG.
355 : A4 06       LDY  $06       VTAB dans Y.
357 : 20 01 E3    JSR  $E301     SGNFLT rend flottant l'entier de Y.
35A : 20 82 E9    JSR  $E982     FMULTT multiplie FAC par ARG (résul. en FAC).
35D : 4C 52 E7    JMP  $E752     GET ADR transforme FAC en 2 octets. RETOUR.
```

BIT À 1

Question : Je n'ai pas bien compris comment on «force un bit à 1»... expression souvent employée dans les commentaires des programmes en Assembleur.

Vous savez qu'un octet comprend 8 bits que l'on représente habituellement sous cette forme (c'est ici la valeur 0 qui est prise en compte) :

0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0

Pour forcer un bit à 1, on utilise l'instruction ORA. Voici quelques exemples :

```
$03      0 0 0 0 0 0 1 1
ORA $80  1 0 0 0 0 0 0 0
= $83    1 0 0 0 0 0 1 1
```

```
$8F      1 0 0 0 1 1 1 1
ORA $20  0 0 1 0 0 0 0 0
= $AF    1 0 1 0 1 1 1 1
```

ECRLETTRES/L1

Programme similaire au précédent, mais un test permet de stopper l'affichage quand l'écran est complet.

```
300 :  20 58 FC    JSR  $FC58
303 :  A9 18       LDA  £$18
305 :  85 06       STA  $06
307 :  20 67 03    JSR  $0367
30A :  98          TYA
30B :  20 47 F8    JSR  $F847
30E :  A9 28       LDA  £$28
310 :  85 06       STA  $06
312 :  20 67 03    JSR  $0367
315 :  98          TYA
316 :  18          CLC
317 :  65 26       ADC  $26
319 :  85 26       STA  $26
31B :  90 02       BCC  $031F
31D :  E6 27       INC  $27
31F :  A9 1A       LDA  £$1A
321 :  85 06       STA  $06
323 :  20 67 03    JSR  $0367
326 :  C8          INY
327 :  98          TYA
328 :  48          PHA
329 :  A0 00       LDY  £$00
32B :  B1 26       LDA  ($26),Y
32D :  C9 80       CMP  £$80
32F :  B0 04       BCS  $0335
331 :  68          PLA
332 :  09 C0       ORA  £$C0
334 :  48          PHA
335 :  68          PLA
336 :  91 26       STA  ($26),Y
338 :  A8          TAY
339 :  2C 30 C0    BIT  $C030
33C :  88          DEY
33D :  10 FA       BPL  $0339
33F :  A9 17       LDA  £$17
341 :  85 06       STA  $06
343 :  A5 06       LDA  $06
345 :  20 47 F8    JSR  $F847
348 :  A0 27       LDY  £$27
34A :  B1 26       LDA  ($26),Y
34C :  C9 A0       CMP  £$A0
34E :  F0 09       BEQ  $0359
350 :  88          DEY
351 :  10 F7       BPL  $034A
353 :  C6 06       DEC  $06
355 :  10 EC       BPL  $0343
357 :  80 0D       BRA  $0366
359 :  2C 10 C0    BIT  $C010
35C :  AD 00 C0    LDA  $C000
35F :  2C 10 C0    BIT  $C010
362 :  C9 1B       CMP  £$1B
364 :  D0 9D       BNE  $0303
366 :  60          RTS
```

RND

```
367 :  A9 01       LDA  £$01
369 :  20 B1 EF    JSR  $EFB1
36C :  20 63 EB    JSR  $EB63
36F :  A4 06       LDY  $06
371 :  20 01 E3    JSR  $E301
374 :  20 82 E9    JSR  $E982
377 :  4C 52 E7    JMP  $E752
```

NOP Cette instruction (inexistante dans les routines) est muette, mais sa durée est de deux cycles et peut allonger des boucles de délai.

BLOLIB.A

Avec ProDOS, il suffit d'afficher le catalogue de la disquette pour connaître le nombre de BLOCKS FREE... mais il peut se révéler intéressant, dans le cadre d'un programme, avant d'enregistrer des données, d'obtenir ce même renseignement sans pour autant passer par le catalogue... d'où cette routine.

```
300:  20 00 BF    JSR  $BF00      Appel classique du MLI.
303:  80                          Commande READ BLOCK.
304:  03 32                       Adresse de la liste des paramètres.
306:  A9 00       LDA  £$00       Nous allons utiliser deux adresses de la
308:  85 06       STA  $06        page 0 pour y écrire le nombre de blocs
30A:  85 07       STA  $07        disponibles.
30C:  A0 C8       LDY  £$C8       Y prêt.
30E:  B9 FF 3F    LDA  $3FFF,Y    Lecture d'un octet.
311:  A2 07       LDX  £$07
313:  0A          ASL             On passe les 8 bits en revue.
314:  90 06       BCC  $031C      Quand un bit est à 0, le bloc est occupé.
316:  E6 06       INC  $06        Quand il est à 1, il est libre.
318:  D0 02       BNE  $031C      ASL pousse chaque bit dans la retenue et il
31A:  E6 07       INC  $07        suffit d'incrémenter quand celle-ci est à 1.
31C:  CA          DEX
31D:  10 F4       BPL  $0313
31F:  88          DEY             Jusqu'à Y = 1 inclus.
320:  D0 EC       BNE  $030E
322:  A9 16       LDA  £$16       Curseur au début de la ligne choisie.
324:  20 5B FB    JSR  $FB5B
327:  20 42 FC    JSR  $FC42      CLREOP efface jusqu'au bas de l'écran.
32A:  A6 06       LDX  $06        LINPTR affiche les deux octets de X et A
32C:  A5 07       LDA  $07        (mode décimal).
32E:  20 24 ED    JSR  $ED24
331:  60          RTS             Retour.
332:  03                          Table des paramètres, pour ProDOS... qui en attend 3 :
333:  60                          1. DRIVE et SLOT $60 pour 6, $50 pour 5, etc. (voir
                                     pour DRIVE la routine BLOLIB.V page suivante).
334:  00 40                       2. Adresse du Buffer : ici $4000
336:  06 00                       3. Numéro du Bloc à lire : ici 6 (BIP.MAP).
```

BSAVE BLOLIB.A,A$300,L$38

BLOLIB.V

Appel par CALL 768, S, D, X... où S est le numéro du SLOT, D celui du DRIVE et X la variable qui, au retour, contiendra le nombre de BLOCKS FREE...

```
300 : 20 F5 E6    JSR $E6F5      GETBYTC transmet (dans X) le numéro du SLOT.
303 : 8A          TXA            On le passe dans A.
304 : 0A          ASL
305 : 0A          ASL            Où 6 devient $60, 5... $50, etc.
306 : 0A          ASL
307 : 0A          ASL
308 : 48          PHA            SLOT sur la pile.
309 : 20 F5 E6    JSR $E6F5      Même opération pour le numéro de DRIVE.
30C : 8A          TXA
30D : 4A          LSR            1 ou 0 tombe dans la retenue.
30E : B0 04       BCS $0314      Si c'est un, rien à faire.
310 : 68          PLA
311 : 09 80       ORA £$80       Si c'est 0, il faut forcer le bit 0 de la valeur
313 : 48          PHA            SLOT à 1.
314 : 68          PLA
315 : 8D 4C 03    STA $034C      $60 devient $E0 (0110 0000... 1110 0000).
318 : 20 00 BF    JSR $BF00      Installation dans la liste des paramètres.
31B : 80                         Appel du MLI.
31D : 4B 03                      $80 : READ Block.
                                 Adresse des paramètres.
31E : A9 00       LDA £$00
320 : 85 06       STA $06        Préparation.
322 : 85 07       STA $07
324 : A0 C8       LDY £$C8       Y prêt.
326 : B9 FF 3F    LDA $3FFF,Y
329 : A2 07       LDX £$07
32B : 0A          ASL
32C : 90 06       BCC $0334
32E : E6 06       INC $06
330 : D0 02       BNE $0334      Voir BLOLIB.A
332 : E6 07       INC $07
334 : CA          DEX
335 : 10 F4       BPL $032B
337 : 88          DEY
338 : D0 EC       BNE $0326
33A : A4 06       LDY $06        GIVAYF rend flottant l'entier relatif dans
33C : A5 07       LDA $07        A,Y (nbre de BLOCS). Résultat dans FAC.
33E : 20 F2 E2    JSR $E2F2
341 : 20 BE DE    JSR $DEBE      Teste la dernière virgule.
344 : 20 E3 DF    JSR $DFE3      Recherche la variable (son adresse est dans A,Y).
347 : AA          TAX            A passe dans X.
348 : 4C 2D EB    JMP $EB2D      MOVMF transfère FAC dans la variable pointée par X et Y.
34B : 03                         Liste des paramètres.
34C : 60                         (Voir BLOLIB.A).
34D : 00 40
34F : 06 00
```

BSAVE BLOLIB.V,A$300,L$51 ■

MLI

Cette routine ne présenterait qu'un intérêt secondaire si elle n'aidait pas le néophyte à comprendre comment utiliser le MLI (Machine Language Interface), partie centrale de PRODOS.SYSTEM. On remarquera que l'affichage des éventuelles erreurs n'y est pas traité.

PRÉPARATION

```
300 : 20 58 FC    JSR FC58
303 : A9 5C       LDA $5C
305 : A0 03       LDY £03
307 : 20 3A DB    JSR DB3A
30A : 20 6F FD    JSR FD6F
30D : 8A          TXA
30E : F0 F0       BEQ 0300
310 : 8D 9A 03    STA 039A
313 : BD 00 02    LDA 0200,X
316 : 29 7F       AND £7F
318 : 9D 9B 03    STA 039B,X
31B : CA          DEX
31C : 10 F5       BPL 0313
```

HOME efface l'écran.

Adresse de la chaîne "TITRE: ".

STROUT affiche cette chaîne.

Entrée de GETLIN sans astérisque.

Le nbre de caractères est dans X. On le passe dans A.

S'il est nul, retour à la saisie.

Longueur du mot stockée (voir plus loin).

Chaîne installée où le MLI saura la lire (grâce à la liste des paramètres).

GET FILE INFO ($C4)

```
31E : 20 00 BF    JSR BF00
321 : C4          C4
322 : 78 03       0378
324 : B0 DA       BCS 0300
326 : A9 64       LDA £64
328 : A0 03       LDY £03
32A : 20 3A DB    JSR DB3A
32D : AE 7D 03    LDX 037D
330 : AD 7E 03    LDA 037E
333 : 20 24 ED    JSR ED24
336 : A9 6E       LDA £6E
338 : A0 03       LDY £03
33A : 20 3A DB    JSR DB3A
33D : AE 80 03    LDX 0380
340 : AD 81 03    LDA 0381
343 : 20 24 ED    JSR ED24
```

Saut au point d'entrée du MLI.

$C4 : code de GET FILE INFO.

Adresse de la liste des paramètres.

Si retenue à 1, erreur et retour au point de départ.

Adresse de la chaîne... "Ad/Long : "

Aussitôt affichée par STROUT.

On lit ici l'adresse de chargement d'un fichier binaire (longueur si c'est un fichier texte).

LINPTR affiche cette adresse ou longueur en (déc.).

Adresse de la chaîne "Blocs: ".

Aussitôt affichée par STROUT.

Nombre de blocs retrouvé dans la liste des paramètres... et affiché par LINPTR.

OPEN ($C8)

```
346 : 20 00 BF    JSR BF00
349 : C8          C8
34A : 8A 03       038A
34C : 90 01       BCC 034F
34E : 60          RTS
```

Nouvel appel au MLI.

$C8 est le code de OPEN (ouvrir le fichier).

Adresse de la liste des paramètres d'OPEN (3).

Si retenue à 0 (pas d'erreur), on saute.

Terminé.

READ ($CA)

```
34F :  20  00  BF      JSR  BF00
352 :  CA              CA
353 :  90  03          0390
```

Nouvel appel au MLI.
$CA est le code de READ (on va lire 512 octets maxi).
Adresse de la liste des paramètres de READ (4).

CLOSE ($CC)

```
355 :  20  00  BF      JSR  BF00
358 :  CC              CC
359 :  98  03          0398
35B :  60              RTS
```

Dernier appel au MLI.
$CC est le code de CLOSE (fermeture de fichier).
Adresse de l'unique paramètre.
Terminé.

CHAÎNES

```
35C :  54 49 54 52 45 3A 20 00          : TITRE: •
364 :  41 64 2F 4C 6F 6E 67 3A 20 00    : Ad/Long: •
36E :  20 20 42 6C 6F 63 73 3A 20 00    : ..Blocs: •
```

PARAMÈTRES GET FILE INFO

```
378 :  0A
379 :  9A  03
37B :  00
37C :  00
37D :  00  00
37F :  00
380 :  00  00
382 :  00  00
384 :  00  00
386 :  00  00
388 :  00  00
```

Nombre de paramètres (10).
Nom d'accès (adresse).
On lira ici le code d'accès.
Le type de fichier.
Le type auxiliaire (adresse de chargement ou longueur).
Le type sauvegarde.
Nombre de blocs occupés.
Date modifiée.
Heure modifiée.
Date de création.
Heure de création.

PARAMÈTRES OPEN

```
38A :  03
38B :  9A  03
38D :  00  20
38F :  00
```

Nombre de paramètres (3).
Adresse du nom d'accès.
Buffer (ici $2000).
Numéro de référence.

PARAMÈTRES READ

```
390 :  04
391 :  01
392 :  00  20
394 :  00  00
396 :  00  00
```

Nombre de paramètres (4).
Numéro de référence.
Adresse du buffer.
Nombre d'octets à lire.
Nombre d'octets lus.

PARAMÈTRES CLOSE

```
398 :  01  01
```

Un seul paramètre : le numéro de référence.

STOCKAGE DE LA CHAÎNE-TITRE

```
39A :  00  00      etc. jusqu'à $3CF
```

Le premier octet contiendra la longueur de la chaîne, les suivants ses caractères ASCII (bit 7 à 0).

Pour essayer cette routine, cherchez un titre dans votre catalogue de disquette (MLI par exemple) et tapez-le à la suite de titre. Normalement, si vous faites ensuite CALL - 151 puis 2000L, vous obtiendrez la liste du programme MLI. Vous pourrez ensuite essayer avec un fichier TXT et, à la même adresse, vous aurez les 512 premiers octets de ce fichier... Après utilisation, amusez-vous à contrôler les résultats dans la liste des paramètres (Adresse ou longueur en $37D.37E, nombre de blocs en $380.381... par exemple).

TAB

Tabulation valable à l'écran et sur imprimante. Attention ! on n'interdit pas les éventuels dépassements (chaîne trop longue, tabulation impossible).

```
100  D$ = CHR$ (4): PRINT D$"PR£3": PRINT : HOME : GOSUB 240
110  FOR I = 1 TO 6: READ A$(I): NEXT
120  FOR I = 1 TO 6: READ B$(I): NEXT
130  X = X + 1: IF X = 1 THEN PRINT : VTAB 1: GOTO 150
140  X = 0: PRINT : VTAB 12
150  INPUT "TABULATION:  ";T: PRINT : IF T < 19 THEN T = 19
160  GOSUB 230
170  VTAB 23: PRINT "(E)NCORE  (M)ENU DISQUETTE  (I)MPRIMANTE
     (A)PPLESOFT  ";: GET R$: VTAB 22: PRINT
180  IF R$ = "M" THEN PRINT D$"RUN MENU"
190  IF R$ = "E" THEN 130
200  IF R$ = "I" THEN PRINT D$"PR£1": PRINT CHR$ (9)"80N": GOSUB
     230: PRINT D$"PR£0"
210  IF R$ <> "A" THEN 170
220  HOME : END
230  FOR I = 1 TO 8: PRINT : CALL 768,A$(I),T,B$(I): NEXT : PRINT : RETURN
240  FOR I = 768 TO 824: READ R: POKE I,R: NEXT : RETURN
250  DATA  169,0,133,9,32,190,222,32,227,223,160,2,177,131,153,6,0,136,
     16,248,200,196,6,240,10,177,7,9,128,32,237,253,24,144,241,165,9,
     240,1,96,32,245,230,138,56,229,6,170,32,74,249,169,1,133,9,208,203
260  DATA Je suis,Tu es,Il ou elle est,Nous sommes,Vous êtes,Ils ou elles
     sont,J'ai,Tu as,Il ou elle a,Nous avons,Vous avez,Ils ou elles ont
```

EXEMPLE :

Je suis	J'ai
Tu es	Tu as
Il ou elle est	Il ou elle a
Nous sommes	Nous avons
Vous êtes	Vous avez
Ils ou elles sont	Ils ou elles ont

```
300 :  A9 00      LDA  £$00     ┐ 0 dans $9.
302 :  85 09      STA  $09      ┘
304 :  20 BE DE   JSR  $DEBE    CHKCOM teste la virgule.
307 :  20 E3 DF   JSR  $DFE3    PTRGET recherche la variable ou la crée.
30A :  A0 02      LDY  £$02     ┐
30C :  B1 83      LDA  ($83),Y  │ Stockage de l'adresse en $7 et 8, de la
30E :  99 06 00   STA  $0006,Y  │ longueur en $6.
311 :  88         DEY           │ A la sortie de la boucle, Y=$FF.
312 :  10 F8      BPL  $030C    ┘
314 :  C8         INY           Y = Y + 1.
315 :  C4 06      CPY  $06      ┐ Quand Y sera égal à la longueur, saut.
317 :  F0 0A      BEQ  $0323    ┘
319 :  B1 07      LDA  ($07),Y  Lecture du caractère.
31B :  09 80      ORA  £$80     Bit 7 à 1.
31D :  20 ED FD   JSR  $FDED    COUT affiche ou imprime le caractère.
320 :  18         CLC           ┐ GOTO 314.
321 :  90 F1      BCC  $0314    ┘
323 :  A5 09      LDA  $09      ┐ S'il y a 0 dans 9, saut.
325 :  F0 01      BEQ  $0328    ┘
327 :  60         RTS           RETOUR.
328 :  20 F5 E6   JSR  $E6F5    ┐ GETBYTC saute un caractère et GETBYT
32B :  8A         TXA           │ évalue l'expression (résultat dans X que
                                ┘ l'on passe dans A).
32C :  38         SEC           Retenue à 1 pour soustraire.
32D :  E5 06      SBC  $06      Moins la longueur de la chaîne.
32F :  AA         TAX           Résultat dans X.
330 :  20 4A F9   JSR  $F94A    PRBL2 envoie X expaces.
333 :  A9 01      LDA  £$01     ┐ $1 dans 9.
335 :  85 09      STA  $09      ┘
337 :  D0 CB      BNE  $0304    Suite.                                  ■
```

ASCII.BCD

La conversion ASCII/BCD d'un nombre est simple puisqu'il est codé, suivant la parité 0-9, 30-39 ou B0-B9. On voit immédiatement qu'il suffit de masquer le quartet de gauche (4 bits) pour obtenir la conversion . (Se lance par **BRUN ASCII.BCD**).

```
300 :  20 58 FC   JSR  $FC58    HOME nettoie l'écran.
303 :  20 0C FD   JSR  $FD0C    RDKEY : curseur clignotant pour 1 caractère.
306 :  C9 9B      CMP  £$9B     ┐ Si c'est ESCAPE, terminé.
308 :  F0 11      BEQ  $031B    ┘
30A :  C9 B0      CMP  £$B0     ┐ Si plus petit que le 0, à refaire.
30C :  90 F5      BCC  $0303    ┘
30E :  C9 BA      CMP  £$BA     ┐ Si plus grand ou égal $BA, à refaire.
310 :  B0 F1      BCS  $0303    ┘
312 :  29 0F      AND  £$0F     AND 0000 1111.
314 :  85 06      STA  $06      Sauvegarde pour récupération éventuelle.
316 :  20 DA FD   JSR  $FDDA    PRBYTE affiche le contenu HEXA de A.
319 :  80 E8      BRA  $0303    On continue.
31B :  60         RTS           Fin.                                    ■
```

POPFT

POMME OUVERTE... POMME FERMÉE... quelles touches ont été pressées ? La première et (ou) la seconde + quel caractère ?
Petite routine à développer.

```
300 : A9 03     LDA £$03    ┐ PR£3.
302 : 20 95 FE  JSR $FE95   ┘
305 : 8D 0D C0  STA $C00D     Affichage 80 colonnes à 1.
308 : A9 91     LDA £$91    ┐ CTRL-Q, pour revenir en 40 colonnes.
30A : 20 ED FD  JSR $FDED   ┘
30D : 20 3A FF  JSR $FF3A     Bell émet son... son.
310 : A9 09     LDA £$09    ┐ $9 est stocké dans $25, puis $FC22 déplace
312 : 20 5B FB  JSR $FB5B   ┘ le curseur en CV ($25).
315 : 2C 10 C0  BIT $C010     CLEAR KEKBOARD.
318 : A9 BE     LDA £$BE    ┐
31A : A2 02     LDX £$02    ┘ PRBL3 va afficher > + 1 espace.
31C : 20 4C F9  JSR $F94C   ┘
31F : AD 00 C0  LDA $C000     Lecture du clavier.
322 : 10 FB     BPL $031F     Retour si aucune touche n'a été pressée.
324 : 48        PHA           A empilé.
325 : 20 9C FC  JSR $FC9C     CLREOL efface jusqu'au bout de la ligne.
328 : 68        PLA           A récupéré.
329 : C9 9B     CMP £$9B      Est-ce ESCAPE ?
32B : D0 01     BNE $032E     Non : on continue.
32D : 60        RTS           Oui, c'est terminé.
32E : A2 02     LDX £$02    ┐ CARACTÈRE ASCII + espace.
330 : 20 4C F9  JSR $F94C   ┘
333 : AD 61 C0  LDA $C061   ┐ Si la touche PO n'a pas été pressée, saut.
336 : 10 04     BPL $033C   ┘
338 : A9 C1     LDA £$C1    ┐ Autrement, voici POMME OUVERTE
33A : D0 07     BNE $0343   ┘ dans A. Saut.
33C : AD 62 C0  LDA $C062   ┐ Si la touche PF n'a pas été pressée, saut.
33F : 10 20     BPL $0361   ┘
341 : A9 C0     LDA £$C0      Sinon, POMME FERMÉE dans A.
343 : 48        PHA           Valeur de A entassée.
344 : 20 80 FE  JSR $FE80     SETINV : affichage en mode inverse.
347 : A9 9B     LDA £$9B    ┐ PRINT ESCAPE.
349 : 20 ED FD  JSR $FDED   ┘
34C : 68        PLA           A récupéré.
34D : 20 ED FD  JSR $FDED     PRINT l'icône.
350 : 48        PHA           A entassé.
351 : A9 98     LDA £$98    ┐ Pour ne plus afficher les icônes
353 : 20 ED FD  JSR $FDED   │ SETNORM pour affichage mode normal.
356 : 20 84 FE  JSR $FE84   ┘
359 : 20 F4 FB  JSR $FBF4     ADVANCE avance le curseur.
35C : 68        PLA           A récupéré.
35D : C9 C0     CMP £$C0      Est-ce PF ?
35F : D0 DB     BNE $033C     Non : ça repart.
361 : 20 62 FC  JSR $FC62     CR envoie son retour chariot.
364 : 4C 0D 03  JMP $030D     Et ça repart.                    ■
```

VERT

Ecriture verticale ? Facile avec cette petite routine ! Trois paramètres : VTAB, HTAB et la variable. Naturellement, la longueur de la variable ne peut dépasser 24 et elle doit être égale ou inférieure à VTAB.
Vous pouvez aussi afficher des espaces, bien entendu. Comment afficher le mot TOTO, contenu dans T$, à partir de la ligne 7, colonne 9 ? En envoyant la séquence CALL 768, 7+4-1, 9, T$... (VTAB + LEN(T$) - 1).

```
100  TEXT : NORMAL : HOME : GOSUB 155
105  VTAB 19: PRINT "TEXTE (maxi = 24 caractères):"
110  VTAB 20: FOR I = 1 TO 24: PRINT ".";: NEXT : PRINT
115  VTAB 19: PRINT "TEXTE (maxi = 24 caractères):"
120  INPUT "";T$: IF T$ = "" THEN 135
125  CALL 768,24,40,T$
130  GOTO 110
135  VTAB 19: CALL - 958: PRINT "(M)ENU DE DISQUETTE (A)PPLE-
     SOFT ";: GET R$
140  IF R$ = "M" OR R$ = "m" THEN PRINT CHR$ (4)"RUN MENU"
145  IF R$ = "A" OR R$ = "a" THEN HOME : END
150  PRINT : GOTO 135
155  FOR I = 768 TO 827: READ A: POKE I,A: NEXT : RETURN
160  DATA 32,245,230,134,6,32,245,230,202,134,7,32,190,222,32,227,
     223,160,2,177,131,153,24,0,136,16,248,201,25,176,28,58,197,6,
     176,23,198,24,198,6,165,6,32,71,248,164,24,177,25,164,7,9,128,
     145,38,198,24,16,235,96
```

```
300:  20 F5 E6    JSR  $E6F5   ┐
303:  86 06       STX  $06     ┘  VTAB mémorisé dans $6.
305:  20 F5 E6    JSR  $E6F5   ┐
308:  CA          DEX          │  HTAB - 1 mémorisé dans $7.
309:  86 07       STX  $07     ┘
30B:  20 BE DE    JSR  $DEBE   ┐  La variable est prise en charge.
30E:  20 E3 DF    JSR  $DFE3   ┘
```

(suite page 94)

```
311 : A0 02      LDY £$02
313 : B1 83      LDA ($83),Y
315 : 99 18 00   STA $0018,Y
318 : 88         DEY
319 : 10 F8      BPL $0313
31B : C9 19      CMP £$19
31D : B0 1C      BCS $033B
31F : 3A         DEC
320 : C5 06      CMP $06
322 : B0 17      BCS $033B
324 : C6 18      DEC $18
326 : C6 06      DEC $06
328 : A5 06      LDA $06
32A : 20 47 F8   JSR $F847
32D : A4 18      LDY $18
32F : B1 19      LDA ($19),Y
331 : A4 07      LDY $07
333 : 09 80      ORA £$80
335 : 91 26      STA ($26),Y
337 : C6 18      DEC $18
339 : 10 EB      BPL $0326
33B : 60         RTS
```

On va avoir l'adresse d'icelle en $19-1A et sa longueur en $18.
A la sortie de la boucle, la longueur est encore dans A.

Si la longueur est égale ou supérieure à $19 (24) : sortie.

A = A - 1.

Si elle est égale ou supérieure au contenu de $6 : sortie itou.

Longueur = longueur - 1 (car on va partir de 0).

VTAB = VTAB - 1. On le met dans A (ne pas oublier que l'on compte de 0 à $17).

GBASCAL fournit l'adresse de la ligne dans $26-27.

Position du caractère dans Y.

Lecture du caractère.

HTAB dans Y.

Bit 7 à 1 pour affichage.

POKE.

Longueur (qui est devenue position du caractère) décrémentée.

Suite jusqu'à 0 inclus.

Retour au Basic. ■

Fonctionne aussi en mode direct.

Achevé d'imprimer
sur les presses de l'imprimerie CITÉ-PRESS
4, rue de la Cour des Noues — 75020 PARIS
Tél. : (1) 46.36.85.10
Dépôt légal : Novembre 1987

N° d'impression : 252
N° d'édition : 625
N° d'ISBN : 2-901124-26-7