ISSN 0753 2806

tremplin micro

Bruitage sur Apple II*

Gestion de menus sous ProDOS

Vos débuts avec le GS Basic !

Multi-images sur le GS

Vitamines C

Illustration J.G. Couteau

M 1631 - 18 - 33,00 F

3791631033006 00180

N° 18 - Bimestriel - Troisième année
5 Janvier - 4 Mars 1988
241 FB - 11 FS - 33 F

JEU sur Apple

Rafi DERYEGHIYAN

vous propose de pénétrer dans...

Le monde des sorciers

... Ceux-ci ont en effet envahi la planète et vous seul pouvez les arrêter. Arriverez-vous à être à la hauteur de la tâche qui vous attend ?

Le programme comporte un véritable analyseur syntaxique. Vos ordres devront être de la forme verbe + complément (le verbe étant à l'impératif). Vous pourrez insérer n'importe quelle phrase entre le verbe et le complément. De plus, le verbe peut être abrégé à trois lettres et le complément à quatre. Par exemple, les trois phrases ci-dessous ont la même signification :

— entre dans la grotte
— entre grotte
— ent grot

A chaque fois que l'ordinateur vous posera une question du type "où" ou "à qui", répondez uniquement par un nom.

Le programme a été rédigé en BASIC standard et il n'est absolument pas protégé. Trop long pour être publié dans *TREMPLIN MICRO*, nous vous le proposons sous deux formes :

- gratuitement si vous commandez la disquette numéro 18 de *TREMPLIN MICRO*.
- au prix symbolique de 20 F (+ frais d'envoi) si vous le commandez seul.

APPLE GS 1988

Il semble que l'on s'achemine enfin vers le remplacement gratuit des VGC et ROM de la première génération... mais nous n'avons pas obtenu confirmation de cette information au moment où nous mettons sous presse. Il s'agit pourtant d'un événement important, justifiant plus qu'un avis conditionnel dans une revue comme *TREMPLIN MICRO*.

Souhaitons donc que nos lecteurs puissent effectivement, au cours des prochains jours, faire remplacer :

- VGC et ROM de leur Apple GS chéri
- Obtenir la version 3.1 de la disquette Système et, dans la foulée, les versions 1.2 de GS.PAINT et VS.COM.

Sans doute aurons-nous l'occasion d'en reparler. Mon correspondant américain dispose de la nouvelle ROM depuis plusieurs mois...

Nestor.

tremplin micro 18

SOMMAIRE

Avec la collaboration de :

ARGOS, Claude AUBRY, Serge BENYAMIN, Rafi DERYEGHIYAN, Joël DESNOUES, Guy HACHETTE, Stephan HADINGER, Yvan KOENIG, NESTOR, Jean PERROT, Clément RENARD, Emile SCHWARZ.

BIMESTRIEL

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

Tous nos prix sont indiqués TTC.

EDITIONS JIBENA

Direction-Rédaction :
Editions JIBENA
Guy-HACHETTE

La Petite Motte — Senillé
86100 CHÂTELLERAULT.

Téléphone :
49-93-66-66

PUBLICITÉ :
Raymond JULLIEN
(1) 45.75.41.81

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

TREMPLIN MICRO — Bimestriel — C'est une publication des Editions JIBENA, 4, rue de la Cour-des-Noues, 75020 PARIS — S.A. au capital de 3600000 F — Imprimé par CITÉ-PRESS/PARIS — **Service de vente :** Presse-Promotion, tél. : 49.93.65.03 — Dépôt légal à la date de parution — Inscription à la Commission Paritaire des Publications et Agences de Presse : en cours — Directeur de la Publication : Guy-Clément COGNÉ — Diffusion N.M.P.P.

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

DISONS-LE CLAIREMENT : Guy-Hachette fait le point 2

BRUITAGE Jean Perrot, fidèle au Basic Applesoft vous propose un choix impressionnant de sons 3

GESTION DE MENUS EN BASIC avec Serge Benyamin 7

MULTI-IMAGES : Une trouvaille de Joël DESNOUES, sur le GS 14

UN LIVRE CAPITAL de Marcel COTTINI 16
PAS D'ACCORD : coups d'aiguille d'Argos, Y. Kœnig et Nestor 16
FREEZER, une exclusivité TREMPLIN MICRO... sur le GS 17

VITAMINES C Claude Aubry vous le dit :
C facile et vous propose même 19
un JEU DE NIM en langage **C** 23

FORMATAGE 3,5 avec Joël DESNOUES (BASIC + LM) 25
LIT.JONG : Nestor s'amuse à lire les fichiers du JONGLEUR DE MOTS ... 27
SBC.ADC avec Clément Renard et le 65816 32
DIFFÉRENCES : initiation au langage machine avec Nestor 33
LIGNE horizontale instantanée 34
LOCNBR : poursuivez votre initiation LM avec Nestor 35
FOUT : et restez encore avec lui pour comprendre ce que fait FOUT 38
LA DATE SUR LE GS (initiation) avec Clément Renard 41

ACCENT L'un des défauts de jeunesse de l'APPLE *II GS* corrigé par Stephan Hadinger 43

PROGRAMMATION COMPARÉE en GSBasic et en C Deux programmes-clé d'Emile SCHWARZ 55

SPRINT... DE BORLAND... Ça marche très fort ! 60
QUELQUES EXEMPLES (simples) EN GSBasic 61
LE COURRIER D'Yvan Kœnig 63
BIBLIOTHÈQUE INFORMATIQUE 13, 16, 31, 54, 62, 67, 69

Disons-le clairement

J'aimerais partager l'optimisme de notre ami Yvan KŒNIG quant à l'avenir du beau dernier de la gamme Apple *II*, le prometteur *GS*. Les plus fidèles lecteurs de *Tremplin Micro* peuvent pourtant témoigner des efforts que nous avons déjà accomplis pour aider les utilisateurs de cette machine. Nous avons, en effet, été les premiers à lui accorder autant de pages.

C'est tellement vrai que les fans des Ancêtres — comprenez par là les Apple *IIe* et *IIc* — nous ont parfois accusés d'être partiaux (le mot n'est pas trop fort) dans nos jugements. Il est vrai que, dans le même temps, d'autres lecteurs me reprochaient certains (libres) propos qui malmenaient leur nouveau joujou. Les plus objectifs savent heureusement que *Tremplin Micro* et son fondateur ne connaissent que l'intérêt du lecteur-consommateur.

Précision utile : cette nouvelle mise au point n'engage que son signataire... lequel n'a pas de compte personnel à régler.

APPLE

Je n'ai de remerciement à adresser à Apple France que pour l'unique prêt d'un *GS* pendant deux semaines, immédiatement après l'annonce de la sortie de la machine. Respectueux de la vérité, je reconnais avoir également reçu, en deux ans, une demi-douzaine de communiqués de presse, mais seulement après des demandes réitérées : il semble que nos coordonnées disparaissent périodiquement du fichier de nos confrères du service de presse. Passons. Heureusement, nous sommes parfaitement informés sur les projets d'Amstrad, Atari, Victor, Donatec... les autres m'excuseront de ne point les citer. Amusant, non ?

Documentation

Les lecteurs de *Tremplin Micro* supposent en général que les animateurs de leur revue préférée croulent sous les notes du support technique et peuvent systématiquement essayer TOUS les logiciels fonctionnant sur Apple. Tout faux ! Si quelques grands éditeurs de logiciels nous honorent de leur confiance (Version Soft, Borland, Cedic/Nathan, Gribouille), d'autres nous ignorent totalement (Microsoft par exemple et, bien sûr, Apple). Par contre, tous les services de presse des Editeurs de livres (sauf Apple) nous adressent leurs ouvrages.

GSBasic

Vous n'avez pas rêvé : non seulement nous avons bien acheté notre GSBasic, mais nous allons prochainement mettre la version française d'une documentation TRÈS COMPLÈTE à la disposition de celles et de ceux qui désireront s'amuser avec ce super Basic américain. Ainsi, une fois de plus, *Tremplin Micro* aidera ses lectrices et lecteurs à programmer, même si cet objectif paraît illusoire aux inconditionnels du Mac. Je connais encore des milliers de personnes qui programment (le contrôle OJD de *T.M.* en fait heureusement foi).

Et puisque j'en suis au chapitre de la programmation, j'avoue être plutôt content d'avoir incité des centaines de lecteurs à jouer avec le langage machine. Vous avez bien lu : JOUER... car c'est d'abord de cela qu'il s'agit. C'est un spécialiste des revues ludiques qui vous l'affirme : pratiquer l'assembleur ou le C est au moins aussi amusant que de croiser les mots ou d'exercer son intellect sur quelque énigme mathématique. Avis aux amateurs que je n'ai pas encore réussi à convaincre !

L'avenir du GS

C'est évidemment là que vous m'attendez. Y croit-il... comme Nestor et comme Yvan KŒNIG... ou a-t-il été définitivement conquis par quelque IBM et autre Atari ? La pratique de la programmation m'a appris à être pragmatique. Il est évident que l'avenir de l'Apple *II GS* ne dépend que de la (bonne) volonté d'Apple. Un fait positif : le remplacement gratuit de la ROM boguée par une nouvelle version... disponible depuis bientôt six mois (que l'on nous excuse si nous ne l'avons pas signalé dans ces colonnes : nous l'avons appris par la presse américaine, puis par des revendeurs). Pour le reste : mystère et bouche cousue. L'information ne passe plus (est-elle jamais passée ?). On ne nous promet rien, mais on laisse les bruits courir. Tenez ! n'importe quel concessionnaire vous murmurera dans le tuyau de l'oreille qu'il est actuellement plus intéressant d'acheter un MAC PLUS qu 'un *GS*. Ce qui est peut-être vrai... quand on ne recherche qu'un traitement de texte ou un tableur, mais faux pour la plupart des applemaniaques venant de l'Apple *II*.

Puisque vous lisez cette revue, il est évident que vous possédez encore l'une de ces merveilleuses petites machines à l'enseigne de la Pomme et, *GS* ou non, j'en suis heureux pour vous. Je sais que mes vœux de bonne et heureuse année sont autant de vœux de moments plaisants et souvent captivants. Merci de votre attention.

G-H

BASIC | DOS | ProDOS

Jean PERROT

BRUITAGE

27 bruits ou sons divers, accompagnés des indications pour les reproduire facilement (utilise le programme LM LASER)

STARTUP

```
100 TEXT : NORMAL : HOME                                    7F31
110 GOSUB 200: PRINT : HTAB 2
    0                                                       2368
120 PRINT "PROGRAMME ¨B R U I
     T A G E, par Jean PERROT
    "                                                       F084
130 HTAB 30: PRINT "Le 25 Sep
    tembre 1987"                                            AF09
140 GOSUB 210: PRINT : VTAB 1
    3                                                       1177
150 PRINT "-Le programme ´BRU
    ITAGE´ que vous lancerez
    en appuyant sur RETURN co
    ntient": PRINT "27 SONS..
    .qui pourront être utilis
    és pour sonoriser un prog
    ramme BASIC."                                           AB11
160 PRINT : PRINT "-L´utilisa
    teur entendra le SON choi
    si et les indications pou
    r reproduire ce SON lui ¨
    seront données en bas de
    l´écran."                                               0C88
170 PRINT : POKE 1403,10: PRI
    NT "´BRUITAGE´ utilise le
     programme L.M. ´LASER´ (
    A$302,L$60)"                                            2AFC
180 VTAB 22: POKE 1403,40: PR
    INT "RETURN pour lancer ´
    BRUITAGE´ ";: GET H$: PRI
    NT                                                      B618
190 PRINT  CHR$ (4)"RUN BRUIT
    AGE"                                                    A3D2
200 PRINT  CHR$ (4)"PR£3": PR
    INT                                                     ED56
210 FOR I = 1 TO 79: PRINT "-
    ";: NEXT                                                3058
220 RETURN                                                  63B1
```

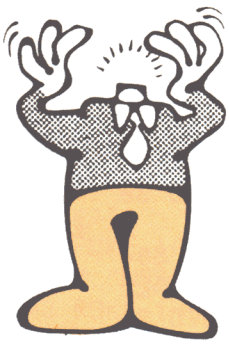
BRUITAGE

```
 20 REM -----(  BRUITAGE  )--
    -----
 21 :                                                       003A
 30 D$ =  CHR$ (4): PRINT D$"
    PR£3": PRINT                                            BD30
 40 IV$ =  CHR$ (15):NR$ =  C
    HR$ (14)                                                449C
 50 PRINT D$"BLOAD LASER"                                   545F
 60 :                                                       003A
 70 HOME : GOSUB 800                                        F319
 80 HOME : POKE 1403,32: PRIN
    T IV$" ¨¨¨¨¨¨¨¨¨¨¨¨¨¨¨¨¨"                               CCFE
 81 POKE 1403,32: PRINT " B R
     U I T A G E "                                          EABD
 82 POKE 1403,32: PRINT " ¨¨¨
    ¨¨¨¨¨¨¨¨¨¨¨¨¨¨"NR$                                     A92E
100 PRINT : PRINT IV$" 1 "NR$
    " Tir ´LASER´ lent"                                     5C75
105 PRINT IV$" 2 "NR$" ¨¨¨¨¨p
    lus rapide"                                             6914
110 PRINT IV$" 3 "NR$" ¨¨¨¨¨t
    res rapide"                                             CF0F
115 PRINT IV$" 4 "NR$" ¨¨¨¨¨e
    ncore plus..                                            6757
```

```
120 PRINT IV$" 5 "NR$" """""..PROUT..."                    93AC
125 PRINT IV$" 6 "NR$" Plusieurs notes basses"             A1B5
130 PRINT IV$" 7 "NR$" """""""""plus hautes"               4BCE
135 PRINT IV$" 8 "NR$" """""""""encore plus"               31C1
140 PRINT IV$" 9 "NR$" Plusieurs notes rapides"            8DFF
145 PRINT IV$" A "NR$" """""""""tres rapides"              AD10
150 PRINT IV$" B "NR$" Laser ascendant"                    9C93
155 PRINT IV$" C "NR$" Laser descendant"                   99DC
160 PRINT IV$" D "NR$" Laser descendant plus court"        13EE
170 PRINT IV$" E "NR$;" PIM-POM (Pompiers)"                D688
180 POKE 1403,40: VTAB 5: PRINT IV$" F "NR$" AMBULANCE "   C653
190 POKE 1403,40: PRINT IV$" G "NR$" MARSEILLAISE"         6F36
195 POKE 1403,40: PRINT IV$" H "NR$" Note longue"          DBDC
200 POKE 1403,40: PRINT IV$" I "NR$" Note courte"          CFE5
205 POKE 1403,40: PRINT IV$" J "NR$" COUCOU"               6D8C
210 POKE 1403,40: PRINT IV$" K "NR$" CUICUI"               F981
215 POKE 1403,40: PRINT IV$" L "NR$" Suite de notes (rapide)"  03C2
220 POKE 1403,40: PRINT IV$" M "NR$" Autre suite (rapide)"     8312
225 POKE 1403,40: PRINT IV$" N "NR$" Suite de notes (lente)"   0C87
230 POKE 1403,40: PRINT IV$" O "NR$" Autre suite (lente)"      A6D7
235 POKE 1403,40: PRINT IV$" P "NR$" LASER 1"              478C
240 POKE 1403,40: PRINT IV$" Q "NR$" LASER 2"              C98E
245 POKE 1403,40: PRINT IV$" R "NR$" Suite de LASERS ascend.et descend."  D1BC
250 POKE 1403,40: PRINT IV$" S "NR$" ""IV$" F I N "NR$"    512F
280 VTAB 19: POKE 1403,65: PRINT IV$"VOTRE CHOIX :"NR$" ";: CALL  - 958: GET H$:H =  ASC (H$): PRINT H$  182F
290 IF H = 13 OR H = 27 THEN VTAB 19: GOTO 280             A3C7
300 IF H < 58 THEN H = H - 48                              5A8C
310 IF H > 64 THEN H = H - 55                              7E85
320 IF H < 1 OR H > 35 THEN VTAB 19: GOTO 280              DA93
330 ON H GOTO 500,510,520,530,540,550,560,570,580,590,600,610,620,630,640,650,660,670,680,690,700,710,720,730,740,750,760,770  DA27
350 POKE 776,A: POKE 777,B: FOR I = 1 TO N: POKE 768,HA: POKE 769,DU: CALL 770: NEXT  0EAF
355 GOSUB 390: GOSUB 380                                   EED1
360 VTAB 22: POKE 1403,65: PRINT IV$"RETURN SVP"NR$;" ";: GET H$  7D4E
370 GOTO 280                                               2345
380 VTAB 21: PRINT "POKE 777,";B;":FOR I=1 TO ";5;":POKE 768,";HA;":POKE 769,";HA;":CALL 770: NEXT": RETURN  AEF6
390 VTAB 20: PRINT "SON obtenu avec progr.B (LASER,A$302,L$60) ";  7EDD
391 IF B = 0 THEN  PRINT " "" """"Faire :""POKE 776,238 "puis "   8AEF
392 IF B = 1 THEN  PRINT " "" """"Faire :""POKE 776,206 "puis "   76EB
395 RETURN                                                 63B1
400 POKE 776,A: POKE 777,B: CALL AD: PRINT " "POKE 777,";B;": CALL ";AD: GOTO 360  9F83
410 POKE 776,A: POKE 777,B: POKE 768,HA: POKE 769,DU: CALL 770: VTAB V  D711
420 PRINT "POKE 777,";B;": POKE 768,";HA;": POKE 769,";DU;": CALL 770": RETURN  FFC4
430 POKE 776,A: POKE 777,B: FOR HA = HD TO HF STEP PAS: POKE 769,DU: POKE 768,HA: CALL 770: NEXT  D635
```

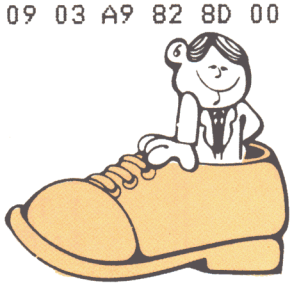
```
431 VTAB 21: PRINT "POKE 777,
    ";B;": FOR HA=";HD;" TO "
    ;HF;" STEP ";PAS;": POKE
    768,HA :POKE 769,";DU;":C
    ALL 770 :"NEXT": GOTO 360   A727
440 POKE 776,A: POKE 777,B: P
    OKE 768,HA: POKE 769,DU:
    CALL 770: FOR K = 1 TO 50
    : NEXT : RETURN             F0B3
445 FOR K = 1 TO 200: NEXT :
    RETURN                      A3C7
446 FOR I = 1 TO 50: NEXT : R
    ETURN                       7198
460 PRINT "      "Voir lignes
     630 et 440 du programme"
    : GOTO 360                  D22B
470 PRINT "      "Voir lignes
     640,641,440,445 du progr
    amme": GOTO 360             F00F
480 PRINT "      "Voir lignes
     650,651,446,950,960 et 9
    70 du programme": GOTO 36
    0                           B75C
500 A = 238:B = 0:HA = 0:DU =
    20:N = 5: GOTO 350          C9FC
510 A = 238:B = 0:HA = 100:DU
    = 100:N = 4: GOTO 350       EE8B
520 A = 238:B = 0:HA = 150:DU
    = 150:N = 4: GOTO 350       EE95
530 A = 238:B = 0:HA = 220:DU
    = 220:N = 6: GOTO 350       2C93
540 A = 238:B = 0:HA = 240:DU
    = 240:N = 6: GOTO 350       5B97
550 A = 206:B = 1:HA = 0:DU =
    240:N = 2: GOTO 350         A429
560 A = 206:B = 1:HA = 100:DU
    = 180:N = 3: GOTO 350       398E
570 A = 206:B = 1:HA = 50:DU
    = 220:N = 4: GOTO 350       5B5E
580 A = 206:B = 1:HA = 200:DU
    = 100:N = 5: GOTO 350       2C89
590 A = 206:B = 1:HA = 220:DU
    = 50:N = 6: GOTO 350        E760
600 A = 238:B = 0:AD = 791: P
    OKE 768,0: GOSUB 390: GOT
    O 400                       A44D
610 A = 238:B = 0:AD = 804: P
    OKE 768,255: GOSUB 390: G
    OTO 400                     B4B4
620 A = 238:B = 0:AD = 821: G
    OSUB 390: GOTO 400          D653
630 A = 206:B = 1:N = 5: FOR
    I = 1 TO N:HA = 100:DU =
    150: GOSUB 440:HA = 130:D
    U = 100: GOSUB 440: GOSUB
    445: NEXT : GOSUB 390: GO
    TO 460                      B1CB
640 A = 206:B = 1:N = 5: FOR
    I = 1 TO N:HA = 130:DU =
    200: GOSUB 440              C6A5
641 HA = 85:DU = 150: GOSUB 4
    40:HA = 100:DU = 100: GOS
    UB 440: GOSUB 445: NEXT :
    GOSUB 390: GOTO 470         2CA4
650 RESTORE : POKE 776,206: P
    OKE 777,1: FOR K = 1 TO 1
    8: READ HA: READ DU         6AD3
651 POKE 768,HA: POKE 769,DU:
    CALL 770: GOSUB 446: NEXT
    : GOSUB 390: GOTO 480       DB20
660 A = 206:B = 1:AD = 817: G
    OSUB 390: GOTO 400          D454
670 A = 206:B = 1:AD = 822: G
    OSUB 390: GOTO 400          5C50
680 A = 206:B = 1:HA = 40:DU
    = 180:V = 21: GOSUB 390:
    GOSUB 410:HA = 50:DU = 20
    0:V = 22: GOSUB 410: GOTO
    360                         6A0F
690 A = 206:B = 1:HA = 15:DU
    = 180:V = 21: GOSUB 390:
    GOSUB 410:HA = 15:V = 22:
    GOSUB 410: GOTO 360         24DD
700 A = 206:B = 1:DU = 30:HD
    = 30:HF = 50:PAS = 2: GOS
    UB 390: GOTO 430            4544
710 A = 206:B = 1:DU = 50:HD
    = 100:HF = 130:PAS = 3: G
    OSUB 390: GOTO 430          A6A4
720 A = 206:B = 1:DU = 220:HD
    = 50:HF = 70:PAS = 6: GOS
    UB 390: GOTO 430            357D
730 A = 206:B = 1:DU = 200:HD
    = 110:HF = 130:PAS = 6: G
    OSUB 390: GOTO 430          ACD5
740 B = 0: CALL 804: CALL 791
    : GOSUB 390: PRINT " CALL
     804 :"CALL 791": GOTO 36
    0                           E996
750 B = 0: CALL 791: CALL 804
    : GOSUB 390: PRINT " CALL
     791 :"CALL 804": GOTO 36
    0                           D296
760 B = 0: FOR I = 1 TO 3: CA
    LL 804: CALL 791: NEXT :
    GOSUB 390: PRINT " FOR I=
    1 TO 3: CALL 804: CALL 79
    1: NEXT": GOTO 360          FBF2
```

```
770 PRINT : VTAB 22: PRINT "C
    'est bien termine ? (O/N)
     ";: GET H$: PRINT H$;          BAF0
780 IF H$ = "O" THEN  POKE 6,
    0: FOR I = 1 TO 50:BZ =
    PEEK (49200): NEXT : POKE
    1403,40: PRINT IV$" VOUS
    AVEZ FINI ? "NR$;"..ALORS
    ...AU REVOIR": END              0E09
790 GOTO 280                        2345
800 REM ======== QUELQUES EXP
    LICATIONS ============
810 POKE 1403,29: PRINT IV$"
    QUELQUES EXPLICATIONS "NR
    $: PRINT                        5C33
820 PRINT "-Ce programme donn
    e 27 notes ou bruitages d
    ifferents :¨LASER descend
    ant, montantpuis des note
    s, suites de notes ETC...
    d'une facon rapide ou plu
    s lente"                        4784
830 PRINT : PRINT "-Quand vou
    s aurez le MENU vous appu
    ierez sur une des touches
     indiquees et vous ¨obtie
    ndrez non seulement le so
    n (ou bruitage) mais auss
    i (en bas de l'ecran)"          6324
840 PRINT "tous les elements
    qui vous permettront de r
    eproduire ce son ou bruit
    age, dans un programme BA
    SIC."                           4E32
870 POKE 1403,25: PRINT "Avec
     le progr. 'LASER' (A$302
    ,L$60) on aura "IV$"UN(E)
    "NR$": PRINT                    6164
880 PRINT IV$"NOTE DE MUSIQUE
    "NR$" si ¨POKE 776,206:PO
    KE 777,1:POKE 768,HA:POKE
     769,DU:CALL 770"               021A
890 POKE 1403,21: PRINT ""En
    faisant varier HA (hauteu
    r de la note) et DU (sa d
    uree)on obtiendra des";:
    PRINT " notes differentes
    .(HA et DU sont inferieur
    s tous deux a 256).";          8833
891 PRINT "Onpourra ainsi con
    struire de petites melodi
    es.§"                           D11A
900 PRINT : PRINT IV$"LASER D
    ESCENDANT"NR$" si ¨¨POKE
    776,238: POKE 777,0: CALL
     770"                           2AE3
901 PRINT : PRINT IV$"LASER A
    SCENDANT"NR$" si ¨¨POKE 7
    76,206:POKE 777,0: CALL 7
    70"                             A076
920 VTAB 21: POKE 1403,40: PR
    INT "RETURN pour le MENU
    ";: GET H$: PRINT               7F87
930 RETURN                          63B1
950 DATA 172,32,172,64,172,32
    ,128,128,128,128,115,128,
    115,128                         DA91
960 DATA 86,128,102,64,128,64
    ,128,32,102,64,128,32,152
    ,128                            F1FA
970 DATA 96,255,115,64,144,32
    ,128,255                        81F8
```



LASER,A$0302,L$0060

```
0302: AD 30 C0 88 D0 05 EE 00 03 F0 09 CA D0 F5          8573
0310: AE 00 03 4C 02 03 60 A9 EE 8D 08 03 A9 00 8D 09    15D0
0320: 03 4C 02 03 A9 CE 8D 08 03 A9 00 8D 09 03 4C 02    D4F3
0330: 03 A9 EE 8D 08 03 A9 01 8D 09 03 A9 6E 8D 00 03    DB1C
0340: A9 40 8D 01 03 4C 02 03 A9 EE A9 EE 8D 08 03 A9    4A3A
0350: 01 8D 09 03 A9 82 8D 00 03 A9 50 8D 01 03 4C 02    DF2D
0360: FF FF                                              FFFE
```



BASIC ProDOS

Serge BENYAMIN

GESTION DE MENUS

Gestion de menus est un module utilisable dans n'importe quel programme BASIC. Il permet de générer des menus sous forme de fiches qui se superposent. Les sélections s'effectuent par déplacement d'un curseur (en mode inverse), grâce aux flèches (validation par la touche **Return**, avec possibilité de revenir, à tout moment, en arrière grâce à la touche **Esc**).

Pour cela, il faut employer deux routines distinctes :

- La première sert à configurer l'arborescence ainsi que le contenu des différents menus.
- La seconde est directement exploitable dans un programme BASIC (elle gère l'arborescence préalablement sauvegardée sous la forme d'un fichier de variables (VAR)).

Configuration :
— Un Apple *II* (+, e, c, GS)
— Minimum 48 Ko
— Prodos (8 ou 16).

Routine de configuration

Il faut d'abord remplir les DATA de la façon suivante :

(Numéro de ligne) **DATA** (Nom de la fiche), (Nom de la première option), (Configuration de la première option), (Nom de la seconde option), (Configuration de la seconde option), ..., **Fin de la fiche.**

(Numéro de ligne)

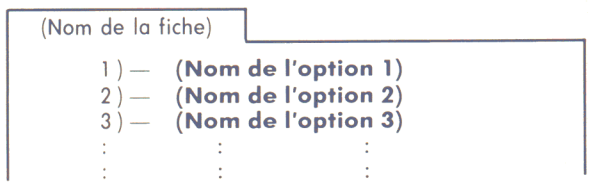
C'est un numéro laissé à votre initiative. *Attention :* chaque fiche déclarée est repérée par un numéro (une fiche commence à la déclaration de son nom et finit quand le programme rencontre *Fin de Fiche*). Ce numéro dépend du nombre de fiches déjà créées. Ainsi, la première fiche aura le numéro 0, la suivante le numéro 1 et ainsi de suite. Ceci est très important pour indiquer l'arborescence des différents menus.

(Nom de la fiche)

Il s'agit du texte qui sera inscrit au début de la fiche.

(Nom de l'option)

Cette zone correspond au libellé inscrit pour chacune des options composant la fiche.



(Configuration de l'option)

On fournit au programme qui va gérer les options les renseignements nécessaires, c'est-à-dire :

- Le numéro du menu précédent (si l'on tape sur (Esc) pour revenir).
- Le numéro du menu suivant. Ainsi, il affichera en le décalant le menu suivant (si il en existe effectivement un).
- Le numéro qui caractérisera l'option choisie. Si aucun menu ne suit, c'est que l'on est arrivé à une option directement exécutable. On retournera alors ce numéro au programme qui a appelé le sous-programme de gestion de menus, et celui-ci pourra agir en conséquence. *(suite page 8)*

Toutes ces informations sont stockées de la manière suivante :

- Les deux premiers chiffres indiquent le numéro qui caractérise l'option (si ce n'est pas une option directement exécutable : ØØ).
- Les deux suivants indiquent le numéro de la prochaine option (si celle-ci n'existe pas : ØØ).
- Les deux derniers correspondent au numéro du menu précédent (comme pour les autres, il faut mettre ØØ s'il n'y a pas de menu qui précède).

Attention : chacun des nombres caractérisant les options comporte obligatoirement deux chiffres : Ø1, Ø9, etc.

Exemples :

1ØØØ DATA Démonstration,Gestion du disque, ØØØ1ØØ,Affichage du catalogue,Ø1ØØØØ,Sortir du programme,Ø2ØØØØ,"Fin de fiche"

1Ø1Ø DATA Gestion du disque,Affichage de la place libre,Ø3ØØØØ,Affichage partiel du catalogue,ØØØ2ØØ,"Fin de fiche"

1Ø2Ø DATA Catalogue,Liste des programmes Basic,Ø4ØØØ1,Liste des fichiers textes,Ø5ØØØ1, "Fin de fiche".

On aura donc :

- un menu principal s'appelant **Démonstration**,
- des options directement exécutables (Affichage du catalogue, Sortir du programme, Affichage de la place libre, Liste des programmes Basic, Liste des fichiers textes) qui donneront à **OP** les valeurs suivantes : Ø1, Ø2, Ø3, Ø4, Ø5,
- des options qui brancheront sur d'autres menus tels que : Gestion du disque, Affichage partiel du catalogue...

Remarque : il est préférable d'encadrer **Fin de fiche** de guillemets afin d'éviter toutes insertions d'espaces qui pourraient perturber le fonctionnement du programme.

L'exécution du programme :

Elle initialise un tableau OP$() à trois dimensions qui sera sauvé sous le nom que vous lui donnerez, sous forme d'un fichier de variables, grâce à l'instruction STORE de PRODOS.

Ce fichier sera ultérieurement exploité par une sous-routine Basic que vous inclurez dans votre propre programme.

CONFIGURATION

```
  100 RESTORE :NL = 0:NC = 0:TT = 0                                          592F
  110 ONERR  GOTO 200                                                        69E2
  120 READ A$:NC = NC + 1                                                    1411
  140 IF NC > TT THEN TT = NC                                                1482
  150 IF A$ = "Fin de fiche" THEN NL = NL + 1:NC = 0                         C0B7
  160 GOTO 120                                                               113E
  200 ONERR  GOTO 300                                                        96E3
  201 DIM OP$(NL - 1, INT (TT / 2) - 1,1)                                    167A
  210 RESTORE :I = 0:J = 1                                                   51B6
  230 READ OP$(I,0,1): REM Initialisation du titre                           6C9D
  240 READ A$                                                                CAEC
  250 IF A$ <  > "Fin de fiche" THEN OP$(I,J,0) = A$: READ OP$(I,J,1):J =
      J + 1: GOTO 240                                                        6626
  260 IF A$ = "Fin de fiche" THEN A$ = "000" +  STR$ (J - 1):A$ =  RIGHT$
      (A$,4):OP$(I,0,0) = A$:J = 1:I = I + 1: GOTO 230                       82BF
  300 INPUT "Nom du fichier a sauvegarder :";A$: PRINT  CHR$ (4);"STORE
      ";A$                                                                   4285
10000 DATA Démonstration,Gestion du disque,000100,Affichage du catalogue,
      010000,Sortir  du programme,020000,"Fin de fiche"                      908F
10010 DATA Gestion du disque,Affichage de la place libre,030000,Affichage
       partiel du catalogue,000200,"Fin de fiche"                            798E
10020 DATA Catalogue,Liste de programes BASIC,040001,Liste des fichiers t
      extes,050001,"Fin de fiche"                                            FCEB
```

Routine gérant les menus

C'est un sous-programme que vous insérerez dans votre propre programme. Vous accomplirez les opérations suivantes :

1. D'abord, avant même d'effectuer vos initialisations de tableaux et de variables, en début de programme, vous devrez recharger le fichier **VAR** que vous aurez configuré préalablement, grâce à RESTORE suivi du nom du fichier.

Exemple : PRINT CHR$(4);"RESTORE OPTIONS"

2. Ensuite, il faudra initialiser les variables, ce qui vous permettra d'adapter le mieux possible ce programme à vos besoins :

ME = 0 : Pour vous positionner sur le premier menu.
HT : Position horizontale du coin gauche.
VT : Position verticale du même point.
F1 : Longueur de la fenêtre.
FH : Hauteur de la fenêtre.

Attention : ce sous-programme utilise les variables suivantes :

TI$: pour le titre, **I**, **PC**, **MA**, **MX**, **CL**, **OP**.

3. Vous n'aurez plus alors qu'à faire un :

GOSUB 41000

Vous récupérerez le numéro de l'option grâce à la variable OP en fonction de laquelle vous pourrez faire des branchements.

Exemple : ON OP GOSUB 1000,2000,3000

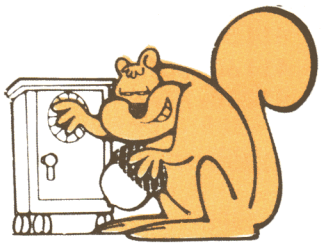
Explications du sous-programme

Il se divise en deux parties :

- De 40000 à 40070, c'est le sous-programme qui affiche (en fonction des paramètres) une fenêtre en forme de fiche. Il pourra ultérieurement être appelé par le programmeur indépendamment de la gestion du menu en sachant que **TI$** contient le titre de la fenêtre. Si ce dernier est une chaîne vide, la fenêtre deviendra un banal rectangle.
- De 41000 à 41130, c'est le sous-programme de gestion qui affiche la fenêtre, les options et permet l'enchaînement des deux et la gestion du curseur.

GESTION DE MENUS

```
   10 REM
   20 REM Programme de gestion de fenêtres
   30 REM Proposé par BENYAMIN Serge
   40 REM
   50 REM 93 Bd de la madeleleine
   60 REM 06000 Nice
   70 REM Tel : 93-86-57-70
   80 REM
39990 REM Sous programme qui affiche
39991 REM le menu
39992 REM On lui passe les renseignements
39993 REM par : HT -> Position Horizontale
39994 REM par : VT -> Position Verticale
39995 REM par : FL -> La longeur de la fenêtre
39996 REM par : FH -> La hauteur de la fenêtre
39997 REM par : TI$ ->Le Titre de la fenêtre
40000 IF  LEN (TI$) = 0 THEN  HTAB HT + 1: VTAB VT + 1: GOTO 40030          81EC
40010 HTAB HT + 1: VTAB VT: FOR I = 1 TO  LEN (TI$) + 2: PRINT "_";: NEXT
      : PRINT                                                              C2E8
40020 HTAB HT: VTAB VT + 1: PRINT "! ";TI$;" !";                           2F21
40030 FOR I = 1 +  LEN (TI$) + 3 * ( LEN (TI$) <  > 0) TO FL: PRINT "_";:
      NEXT                                                                 9C44
40040 FOR I = 2 TO FH - (VT + FH - 21) * (21 <  = VT + FH): HTAB HT: VTAB
      VT + I: PRINT "!"; SPC( FL);"! ""                                    3995
```

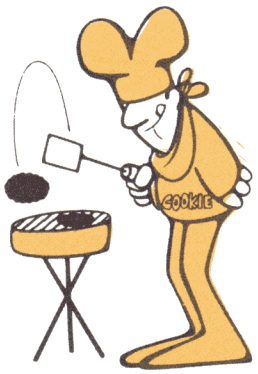
```
40050 NEXT                                                              0582
40060 VTAB VT + FH - (VT + FH - 21) * (21 <  = VT + FH) + 1: HTAB HT: PRI
      NT "!";: FOR I = 1 TO FL: PRINT "_";: NEXT : PRINT "! ": PRINT  SP
      C( 159)                                                           B0FE
40070 RETURN                                                            63B1
40989 REM
40990 REM Il permet d'afficher la suite de menu
40991 REM jusqu'a l'obtention d'un menu exécutable
40992 REM Il a dans OP$(X,Y,Z) toutes les infos nécessaires
40993 REM Il suffit de lui passer par ME le numéro
40998 REM du menu à exécuter
40999 REM
41000 PC =  VAL ( LEFT$ (OP$(ME,0,0),2)):MA =  INT ((FH + 2 - ( VAL ( RIG
      HT$ (OP$(ME,0,0),2)))) / 2) + 1:TI$ = OP$(ME,0,1):MX =  VAL ( RIGHT
      $ (OP$(ME,0,0),2)) - 1: GOSUB 40000                               E83C
41010 FOR I = 0 TO MX: IF I = PC THEN  INVERSE                          C925
41020 HTAB HT + 10: VTAB VT + MA + I: PRINT I + 1;")-";OP$(ME,I + 1,0): N
      ORMAL                                                             D7D5
41030 NEXT                                                              0582
41040 CL =  PEEK (49152): IF CL < 128 THEN 41040                        0336
41050 POKE 49168,0:CL =  PEEK (49152)                                   6EF2
41060 IF CL = 08 OR CL = 11 THEN  GOSUB 41200:PC = PC - 1:PC = PC * (PC >
      = 0) + MX * (PC < 0): GOSUB 41300: GOTO 41040                     D521
41070 IF CL = 10 OR CL = 21 THEN  GOSUB 41200:PC = PC + 1:PC = PC * (PC <
      = MX): GOSUB 41300: GOTO 41040                                    1175
41080 IF CL > 48 AND CL <  = MX + 49 THEN  GOSUB 41200:PC = CL - 49: GOSU
      B 41300: GOTO 41040                                               6FDB
41090 IF (CL = 27 OR CL = 9) AND ME <  > 0 THEN ME =  VAL ( RIGHT$ (OP$(M
      E,PC + 1,1),2)):HT = HT - 2:VT = VT - 2: GOTO 41000               1F7A
41100 IF CL <  > 13 THEN 41040                                          38FD
41110 IF  VAL ( MID$ (OP$(ME,PC + 1,1),3,2)) = 0 THEN OP =  VAL ( LEFT$ (
      OP$(ME,PC + 1,1),2)): RETURN                                      7F3C
41120 IF PC < 10 THEN OP$(ME,0,0) = "0"                                 9BD8
41122 OP$(ME,0,0) = OP$(ME,0,0) +  STR$ (PC)                            D61C
41124 IF MX + 1 < 10 THEN OP$(ME,0,0) = OP$(ME,0,0) + "0"               CE09
41126 OP$(ME,0,0) = OP$(ME,0,0) +  STR$ (MX + 1)                        EF27
41130 IF ME =  VAL ( MID$ (OP$(ME,PC + 1,1),3,2)) THEN 41160            C7B8
41140 ME =  VAL ( MID$ (OP$(ME,PC + 1,1),3,2)):HT = HT + 2:VT = VT + 2  B3DF
41150 GOTO 41000                                                        3EA0
41160 GOTO 41010                                                        38A1
41200 HTAB HT + 10: VTAB VT + MA + PC: PRINT PC + 1;")-";OP$(ME,PC + 1,0)
      : RETURN                                                          9BC7
41300 HTAB HT + 10: VTAB VT + MA + PC: INVERSE : PRINT PC + 1;")-";OP$(ME
      ,PC + 1,0): NORMAL : RETURN                                       4476
```

Vous programmez en Basic : très bien !

Amusez-vous aussi à concevoir de petites routines en langage machine. C'est un jeu absolument passionnant ! Les petits recueils de *TREMPLIN MICRO* (4 volumes parus, avec disquettes) vous fourniront de multiples exemples commentés (bulletin de commande p. 68).

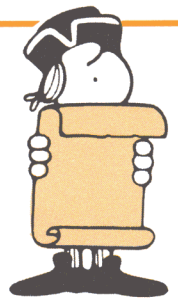
STARTUP-MENU

voici un exemple complet, réalisé en suivant les directives de l'auteur.




```
  10 REM                                                          
  20 REM Programme de gestion de fenêtres                         
  30 REM Proposé par BENYAMIN Serge                               
  40 REM                                                          
  50 REM 93 Bd de la madeleleine                                  
  60 REM 06000 Nice                                               
  70 REM Tel : 93-86-57-70                                        
  80 REM                                                          
 100 PRINT  CHR$ (4);"CLOSE"                                      641B
 110 TEXT : PRINT  CHR$ (4);"PR£3"                                F760
 120 PRINT  CHR$ (4);"RESTORE OPTIONS"                            3015
 125 FL = 65:FH = 11                                              26C7
 270 ME = 0:HT = 2:VT = 2: HOME : PRINT                           BC15
 290 GOSUB 41000                                                  12A5
 300 ON OP GOSUB 1000,2000,3000,4000,5000                         5D82
 310 GOTO 290                                                     2946
 990 REM                                                          
 991 REM Sous programme d'affichage du catalogue                  
 992 REM                                                          
1000 VTAB 1: PRINT  CHR$ (4);"Prefix": INPUT NO$                  85FF
1030 HT = HT + 2:VT = VT + 2:TI$ = "Affichage du catalogue " + NO$: GOSU
     B 40000                                                      04CC
1040 RE$ = "": GOSUB 6000: RETURN                                 EB6A
1990 REM                                                          
1991 REM Sortie du programme                                      
1992 REM                                                          
2000 HOME : HTAB 10: VTAB 10: PRINT "Programme de présentation, Merci et
      au revoir": END                                             745E
2990 REM                                                          
2992 REM Affichage de la place libre sur le disque                
2993 REM                                                          
3000 VTAB 1: PRINT  CHR$ (4);"Prefix": INPUT NO$                  85FF
3030 HT = HT + 2:VT = VT + 2:TI$ = "Affichage de la place libre ": GOSUB
     40000                                                        B13E
3040 PRINT  CHR$ (4);"OPEN ";NO$;",TDIR": PRINT  CHR$ (4);"READ ";NO$:I
     = 0                                                          0A6B
3050 VTAB 1: INPUT A$: IF A$ = "" THEN I = I + 1                  EC75
3060 IF I = 2 THEN 3100                                           B480
3070 GOTO 3050                                                    6173
3100 INPUT A$: HTAB HT + 5: VTAB VT + 5: PRINT "Il vous reste "; VAL ( M
     ID$ (A$,15,3));" Ko sur la disquette";                       2CE7
3110 POKE  - 16368,0                                              40E6
3115 A =  PEEK ( - 16384): IF A < 128 THEN 3115                   4F35
3120 POKE  - 16368,0: PRINT  CHR$ (4);"CLOSE";NO$:HT = HT - 2:VT = VT -
     2: RETURN                                                    84B8
3990 REM                                                          
3991 REM Affichage des programme BASIC                            
3992 REM                                                          
4000 VTAB 1: PRINT  CHR$ (4);"Prefix": INPUT NO$                  85FF
4030 HT = HT + 2:VT = VT + 2:TI$ = "Liste des programmes BASIC": GOSUB 4
     0000                                                         F363
4040 RE$ = "BAS": GOSUB 6000: RETURN                              0F40
4990 REM                                                          
4991 REM Affichage des Fichiers Textes                            
```

```
5000 VTAB 1: PRINT  CHR$ (4);"Prefix": INPUT NO$                                85FF
5030 HT = HT + 2:VT = VT + 2:TI$ = "Liste des Fichiers Texte": GOSUB 400
     00                                                                         93DB
5040 RE$ = "TXT": GOSUB 6000: RETURN                                            686A
5990 REM
5991 REM Module d'affichage du catalogue
5992 REM
 6000 PRINT  CHR$ (4);"OPEN ";NO$;",TDIR": PRINT  CHR$ (4);"READ ";NO$:I
      = 0                                                                       0A6B
 6010 FOR J = 0 TO 2: INPUT A$: NEXT : REM On saute le début                     DF9D
 6020 VTAB 1: INPUT A$: IF A$ = "" THEN 6200                                     1BE2
 6030 IF RE$ = "" THEN 6050                                                      420B
 6040 IF  MID$ (A$,18,3) <  > RE$ THEN 6020                                      E128
 6050 VTAB VT + 3 + I: HTAB VT + 5: PRINT  MID$ (A$,2,15);                       448E
 6060 I = I + 1                                                                  805B
 6070 IF I > FH - 4 THEN 6300                                                    A9DD
 6080 GOTO 6020                                                                  5573
 6190 REM
 6191 REM Si on est a la fin du catalogue
 6192 REM
 6200 VTAB VT + FH: HTAB HT + FL - 10: PRINT " """";                             43A5
 6210 POKE  - 16368,0                                                            40E6
 6220 A =  PEEK ( - 16384): IF A < 128 THEN 6220                                 1135
 6230 POKE  - 16368,0: PRINT  CHR$ (4);"CLOSE";NO$:HT = HT - 2:VT = VT -
      2: RETURN                                                                 84B8
 6290 REM
 6292 REM Si l'on est a la fin de la fenetre
 6293 REM
 6300 VTAB VT + FH: HTAB HT + FL - 10: PRINT "../..";                            5EEC
 6310 POKE  - 16384,0                                                            36E4
 6320 A =  PEEK ( - 16384): IF A < 128 THEN 6320                                 2736
 6330 POKE  - 16368,0: FOR I = 0 TO FH - 4: HTAB HT + 5: VTAB VT + 3 + I:
      PRINT  SPC( 15);: NEXT                                                    376E
 6340 I = 0: VTAB VT + FH: HTAB HT + FL - 10: PRINT " """";: GOTO 6020          65D5
39990 REM Sous programme qui affiche
39991 REM le menu
39992 REM On lui passe les renseignements
39993 REM par : HT -> Position Horizontale
39994 REM par : VT -> Position Verticale
39995 REM par : FL -> La longeur de la fenêtre
39996 REM par : FH -> La hauteur de la fenêtre
39997 REM par : TI$ ->Le Titre de la fenêtre
40000 IF  LEN (TI$) = 0 THEN  HTAB HT + 1: VTAB VT + 1: GOTO 40030              81EC
40010 HTAB HT + 1: VTAB VT: FOR I = 1 TO  LEN (TI$) + 2: PRINT "_";: NEXT
      : PRINT                                                                   C2E8
40020 HTAB HT: VTAB VT + 1: PRINT "! ";TI$;" !";                                2F21
40030 FOR I = 1 +  LEN (TI$) + 3 * ( LEN (TI$) <  > 0) TO FL: PRINT "_";:
      NEXT                                                                      9C44
40040 FOR I = 2 TO FH - (VT + FH - 21) * (21 <  = VT + FH): HTAB HT: VTAB
      VT + I: PRINT "!"; SPC( FL);"! """                                        3995
40050 NEXT                                                                      0582
40060 VTAB VT + FH - (VT + FH - 21) * (21 <  = VT + FH) + 1: HTAB HT: PRI
      NT "!";: FOR I = 1 TO FL: PRINT "_";: NEXT : PRINT "! "": PRINT  SP
      C( 159)                                                                   B0FE
```



```
40070 RETURN                                                          63B1
40989 REM
40990 REM Il permet d'afficher la suite de menu
40991 REM jusqu'à l'obtention d'un menu exécutable
40992 REM Il a dans OP$(X,Y,Z) toutes les infos nécessaires
40993 REM Il suffit de lui passer par ME le numéro
40998 REM du menu à exécuter
40999 REM
41000 PC =  VAL ( LEFT$ (OP$(ME,0,0),2)):MA =  INT ((FH + 2 - ( VAL ( RIG
      HT$ (OP$(ME,0,0),2)))) / 2) + 1:TI$ = OP$(ME,0,1):MX =  VAL ( RIGHT
      $ (OP$(ME,0,0),2)) - 1: GOSUB 40000                             E83C
41010 FOR I = 0 TO MX: IF I = PC THEN  INVERSE                        C925
41020 HTAB HT + 10: VTAB VT + MA + I: PRINT I + 1;")-";OP$(ME,I + 1,0): N
      ORMAL                                                           D7D5
41030 NEXT                                                            0582
41040 CL =  PEEK (49152): IF CL < 128 THEN 41040                      0336
41050 POKE 49168,0:CL =  PEEK (49152)                                 6EF2
41060 IF CL = 08 OR CL = 11 THEN  GOSUB 41200:PC = PC - 1:PC = PC * (PC >
      = 0) + MX * (PC < 0): GOSUB 41300: GOTO 41040                   D521
41070 IF CL = 10 OR CL = 21 THEN  GOSUB 41200:PC = PC + 1:PC = PC * (PC <
      = MX): GOSUB 41300: GOTO 41040                                  1175
41080 IF CL > 48 AND CL <  = MX + 49 THEN  GOSUB 41200:PC = CL - 49: GOSU
      B 41300: GOTO 41040                                             6FDB
41090 IF (CL = 27 OR CL = 9) AND ME <  > 0 THEN ME =  VAL ( RIGHT$ (OP$(M
      E,PC + 1,1),2)):HT = HT - 2:VT = VT - 2: GOTO 41000             1F7A
41100 IF CL <  > 13 THEN 41040                                        38FD
41110 IF  VAL ( MID$ (OP$(ME,PC + 1,1),3,2)) = 0 THEN OP =  VAL ( LEFT$ (
      OP$(ME,PC + 1,1),2)): RETURN                                    7F3C
41120 IF PC < 10 THEN OP$(ME,0,0) = "0"                               9BD8
41122 OP$(ME,0,0) = OP$(ME,0,0) +  STR$ (PC)                          D61C
41124 IF MX + 1 < 10 THEN OP$(ME,0,0) = OP$(ME,0,0) + "0"             CE09
41126 OP$(ME,0,0) = OP$(ME,0,0) +  STR$ (MX + 1)                      EF27
41130 IF ME =  VAL ( MID$ (OP$(ME,PC + 1,1),3,2)) THEN 41160          C7B8
41140 ME =  VAL ( MID$ (OP$(ME,PC + 1,1),3,2)):HT = HT + 2:VT = VT + 2 B3DF
41150 GOTO 41000                                                      3EA0
41160 GOTO 41010                                                      38A1
41200 HTAB HT + 10: VTAB VT + MA + PC: PRINT PC + 1;")-";OP$(ME,PC + 1,0)
      : RETURN                                                        9BC7
41300 HTAB HT + 10: VTAB VT + MA + PC: INVERSE : PRINT PC + 1;")-";OP$(ME
      ,PC + 1,0): NORMAL : RETURN                                     4476
```

LES GUIDES ESSENTIELS : Une collection VIRGA/MARABOUT

Les utilisateurs de PC et compatibles apprécieront certainement les petits guides qui viennent de paraître dans la collection VIRGA/MARABOUT. Ils présentent en effet, sous une forme simple et pratique, mais avec une grande clarté, toutes les commandes utilisées par les langages et progiciels best-sellers de l'informatique.

Un exemple accompagne parfois l'énoncé de la commande. C'est en quelque sorte, ***l'exemple joint au précepte***, comme le disent les publicités d'une célèbre Ecole par correspondance.

Notons, parmi les dernières parutions : MS-DOS, GW.BASIC, TURBO PASCAL et dBASEIII. De très bonnes éditions de poche !

CI. R.

Joël DESNOUES

MULTI-IMAGES

OPTIONS

1. Récupérer l'écran SHGR (pratique en cas d'utilisation de THGS ou en cas de Reset de programmes affichant en SHGR, un reboot par Prodos n'effaçant pas l'écran SHGR contrairement aux écrans HGR qui contraignaient à déplacer l'écran pour ne pas l'écraser lors du reboot).
2. Charger une image (Type $C1 ou BIN et 65 Blocs uniquement).

Après visualisation (ligne 120), vous pouvez :

— soit diviser cette image par quatre et la sauvegarder,

— soit sauver l'image originale.

Pour la division, le principe est simple (lignes 140, 160). Je transfère uniquement les octets correspondant aux pixels de l'écran, sans linéariser (ligne 140). Ensuite, j'ai amélioré, car pour le retour, j'obtenais à l'origine une image dont la première moitié était bonne mais dont la deuxième présentait un décalage. Donc, je transfère cette 1er moitié en E1/2000.E1/5E7F (145.150), puis en E1/5E80.E1/9CFF (160). Ensuite, je supprime les lignes 150 et 160 pour voir la différence.

Ce processus permet de garder intacts la palette et le SCB. Il peut être répété x fois, l'image étant alors divisée par puissance de quatre : 1/4/16/256... mais le résultat devient illisible au delà de 16.

Le reste du programme est la mise en forme, la sauvegarde et le traitement d'erreurs. La routine en LM est "connue" et fait appel uniquement à MVN dont on modifie les paramètres.

```
  1 VID =  PEEK (49193):NV = 65: POKE 6,VID: POKE 49193,NV: GOSUB 500     1D8D
  5 D$ =  CHR$ (4): PRINT  CHR$ (17): ONERR  GOTO 1000                    2983
 10 HOME :T = 5: VTAB T: HTAB T: PRINT "1-ctG Recuperer l'écran SHGR":
    VTAB T + 4: HTAB T: PRINT "2-ctG Charger une image SHGR"              3A57
 20 VTAB T + 8: PRINT "Votre choix -> ";: GET X$                         A0B3
 30 ON X$ = "1" GOTO 120: ON X$ <  > "2" GOTO 255: GOTO 100              9AF0
 40 VTAB T + 3: HTAB T: PRINT "Sauvegarder cette image ? O/N ";: GET X$  9DB8
 50 ON X$ = "O" GOTO 200: GOTO 255                                       2E51
100 PRINT : PRINT : PRINT D$"CAT"                                        5326
105 PRINT : PRINT "Images Type $C1 Ou Bin et 65 Blocs": PRINT : INPUT "
    NomctG --> ";NM$                                                     2A0F
110 T$ = "BIN": ON NM$ = "" GOTO 275                                     3E6F
115 PRINT D$"BLOAD"NM$",T"T$",A$1000": CALL 768                          01C4
120 POKE 49193,161: GET X$: POKE 49193,NV                                F1C8
125 HOME : VTAB T: HTAB T: PRINT "Diviser l'image O/NctG ";: GET X$      31F1
130 ON X$ <  > "O" GOTO 40                                               6F72
140 POKE 49193,1: POKE 781,124: GOSUB 530                                4BF8
145 POKE 49193,97: POKE 775,32: POKE 778,16: POKE 783,225: POKE 784,0    4EFE
150 POKE 780,127: POKE 781,62: CALL 768                                  38B0
160 POKE 774,128: POKE 775,94                                            1E51
165 CALL 768: POKE 49193,161: GET X$: POKE 49193,NV                      5C33
```

```
170 HOME : VTAB T: HTAB T: PRINT "Dingue ...Non ?"                          21D3
175 PRINT : PRINT : PRINT "Sauvegarder cette Version O/NctG ";: GET X$      1BBC
180 ON X$ <  > "O" GOTO 255                                                32AA
200 VTAB 13: PRINT : INPUT "Nom de l'imagectG ";NM$: PRINT : PRINT "Uni
    té de sauvegarde"                                                      5C3D
205 PRINT : INPUT "ctG ¯¯¯¯¯Slot ¯-> ";S$                                  F642
210 PRINT : INPUT "ctG ¯¯¯¯¯Drive -> ";L$                                  AF53
215 RESTORE                                                                5DAE
220 GOSUB 500: GOSUB 530:D$ =  CHR$ (4): PRINT D$"PREFIX": INPUT PFIX$     142C
225 PRINT D$"PREFIX,S"S$",D"L$"                                            5770
230 PRINT D$"CREATE"NM$",T$C1"                                             8935
235 PRINT D$"BSAVE"NM$",T$C1,A$1000,E$8FFF"                                AAE3
240 PRINT D$"CAT"                                                          BD3E
250 PRINT D$"PREFIX"PFIX$                                                  BB8F
255 X =  PEEK (6): POKE 49193,X                                            F512
260 RESTORE : PRINT : PRINT : VTAB 22: PRINT "(1) ENCORE (2) MENU (3) F
    IN ";: GET R$                                                          F049
265 ON  VAL (R$) GOTO 1,270,275                                            71CB
270 REM ***
275 HOME : PRINT "Au revoir...": PRINT : END                               7D94
500 FOR ADR = 768 TO 788: READ J: POKE ADR,J: NEXT : RETURN                1627
510 DATA 139,24,251,194,48,160,0,32,162,0,16,169,255,127,84,225,0          FC4F
520 DATA 56,251,171,96                                                     1912
530 POKE 775,16: POKE 778,32: POKE 783,0: POKE 784,225: CALL 768: RETUR
    N                                                                      C1BB
1000 BUG =  PEEK (222): POKE 222,0: ON BUG = 13 GOTO 1020: ON BUG = 19 G
    OTO 1030                                                               093F
1010 PRINT "Erreur INOPINEE...": END                                       B98E
1020 T$ = "$C1": GOTO 115                                                  12A0
1030 PRINT : PRINT "Cette image existe deja": PRINT : PRINT "Détruire l'
    ancienne O/N ";: GET X$                                                AC65
1040 ON X$ = "O" GOTO 235: GOTO 250                                        1254
```

```
300: 8B          PHB       ;Empile registre Banc de données
301: 18          CLC       ;passage en mode natif
302: FB          XCE
303: C2 30       REP £30   ;Pur A,X,Y sur 16 Bits
305: A0 00 20    LDY £2000 ;AdrL,AdrH Destination
308: A2 00 10    LDX £1000 ;AdrL,AdrH Source
30B: A9 FF 7F    LDA £7FFF ;Nombre d'octets à tranférer
30E: 54 E1 00    MVN 00E1  ;Bank Dest - Bank Source
311: 38          SEC       ;Retour au mode émulation
312: FB          XCE
313: AB          PLB       ;Récupère Banc de données origine1
314: 60          RTS       ;Retour
```

ROUTINE LM incorporée

Pages 25 et 26, vous trouverez aussi une autre batterie de programmes de Joël DESNOUES :

FORMATAGE 3,5 et **CRÉATION DU BLOC 0**

UN LIVRE CAPITAL

Auteur :
Marcel COTTINI

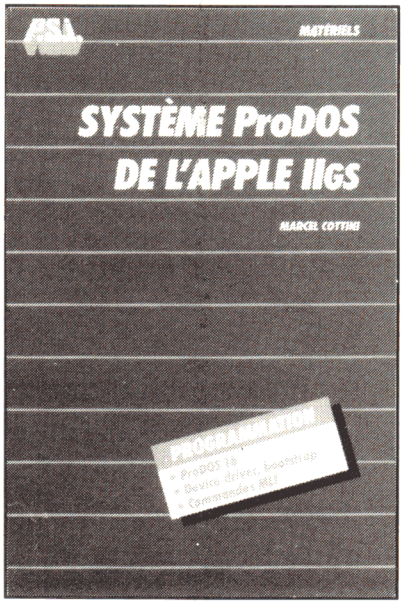
Système ProDOS16 de l'Apple IIGS est un manuel de référence de haut niveau destiné aux programmeurs "hobbyistes" sur Apple *IIGS*. Il brosse un tableau complet des principes de base des systèmes ProDOS8 et ProDOS16 qu'il est souhaitable de connaître, puis s'attaque aux notions élémentaires pour terminer par les commandes MLI. Vous apprendrez, dans un premier temps, à vous familiariser avec les notions fondamentales indispensables à la maîtrise du système, en mettant l'accent sur l'environnement Hardware et Software de ProDOS16. Puis, après avoir détaillé les nouveaux protocoles établis par Apple Computer, vous saurez faire la différence entre les diverses structures de chaque système ProDOS, et vous saurez ce que représente le Boot d'un disque système. Des listings complets et détaillés sont présentés dans ce livre pour la première fois : Reset Hardware Handler, ColdStart, routine complète du BootStrap, et pour finir, la routine de chargement du système ProDOS (système Loader, blocs $00 et $01 du disque).

Enfin, les différentes entrées possibles de chaque table des matières seront développées : entrées d'un Volume Directory Header, d'un Volume SubDirectory Header, et d'un nom de fichier. La gestion des fichiers sera abordée sans détour, mettant l'accent sur le traitement et les différents types de fichiers.

Pour terminer, une large part sera faite aux commandes MLI du système ProDOS16, en traitant chaque commande en détail. Le champ de chaque table des paramètres (Parameter Block) d'une commande vous sera expliqué, et la syntaxe dévoilée.

ProDOS8, système d'exploitation de la génération des processeurs 8 bits, est totalement compatible avec ProDOS 1.1.1.

ProDOS16, système d'exploitation de la nouvelle génération, traite en totalité les instructions du microprocesseur 65C816, vrai 16 bits. Il est particulièrement destiné à l'Apple *IIGS*, dernier-né d'Apple Computer.



Editions du P.S.I.
5, place du Colonel Fabien 75010 PARIS

PAS D'ACCORD

François GALLET a mis plusieurs de nos lecteurs en colère... parce que son AMPERCMD ne respecte pas les règles de ProDOS.
Vrai ou faux ? Si on met 55 END dans le programme Basic et que l'on tape ensuite un RUN, puis un CAT, on ne trouve plus la routine en $8000.
Un remède : s'offrir une ligne 55 HIMEM:31744, mais ce n'est qu'un emplâtre. Le plus simple — c'est le conseil d'Yvan KŒNIG — serait d'utiliser RELOPRO.S, prévu, entre autres choses, pour la mise en place de routines de ce genre. RELOPRO devrait devenir le standard *TREMPLIN MICRO*, même si ce module relogeur occupe près de 2 pages de la mémoire. La vraie contrainte se situe au niveau de l'architecture du programme à reloger, à savoir :

module programme, module adresse (optionnel), module datas (optionnel).

J'ai repris ici une partie des propos tenus par Yvan KŒNIG qui signale au passage que RELOPRO doit beaucoup à Claude AUBRY. Personnellement, je crois que l'auteur n'a pas suffisamment expliqué comment utiliser RELOPRO... à partir d'un exemple très simple. **ARGOS.**

Page 73 de *Tremplin Micro n°15*, l'auteur utilise PLX - TXA alors qu'un simple PLA suffisait. Le fait d'empiler un registre n'oblige pas à dépiler vers le même registre. Attention ! GBASCALC revient avec C=0. **Yvan KŒNIG.**

Que ceux qui possèdent encore le vieux 6502 lèvent la main ! Comment ? Si peu nombreux ? Mais alors, pourquoi se priver plus longtemps d'utiliser les nouvelles instructions du 65C02 ? Ah ! si j'avais le BRA plus long, je vous promets (ça n'engage à rien) que j'aurais décidé Apple à remplacer gratuitement les 6502 par leur jeune successeur. Comme cela, pour le plaisir ! **NESTOR.**

Faux ! dans le "***6502 pas à pas***", les auteurs écrivent que le mini-assembleur est intégré dans le programme moniteur, mais ils se trompent. Il n'est en fait présent que dans des Apple *IIc* récents, capables de reconnaître les disquettes 3 pouces 1/2.

Yvan KŒNIG.

Yvan KOENIG

Une exclusivité Tremplin Micro :

FREEZER

Possesseur d'un GS tout frais, vous vous inquiétez à juste titre du devenir de la carte mère, à la suite des opérations extinction-allumage que vous lui faites subir pour cause de configuration inadéquate du RAMdisk. Vous avez remarqué que la sélection effectuée au tableau de bord ne prend effet qu'après la séquence susdite.

FREEZER vient à votre secours et sera suivi de son quasi homonyme : ***FREEZER.CDA.*** Ce dernier sera utilisable après un démarrage sur *ProDOS16.* Aujourd'hui, je travaille pour ceux — nombreux — qui programment sous ProDOS8.

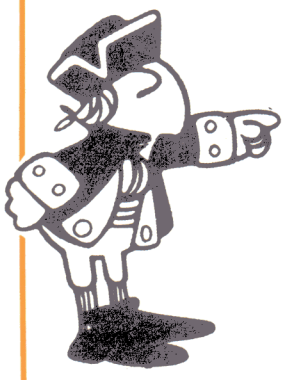
Lors du démarrage, la fonction *MMBootInit du Memory Manager* s'oriente vers un démarrage à froid ou à chaud, en fonction du contenu d'un octet. Comme à peu près tout ce qui est dans le GS, cet octet peut se promener. Il ne l'a pas fait à l'occasion du changement de ROM, mais ça nous pend au nez. J'ai donc choisi de récupérer son adresse dans le code même de l'outil en question, en respectant scrupuleusement les procédures d'approche réglementaires. J'envoie d'ailleurs ***FREEZER*** et ***FREEZER.CDA*** à Rich. Williams, le père du *Memory Manager,* pour lui demander confirmation du fait que la nature de la première instruction de *MMBootInit* restera inchangée (ce qui est probable).

Dans l'état actuel du GS, après un *BLOAD FREEZER,* faire *CALL 768+3* forcera à Ø l'octet de contrôle provoquant un démarrage à froid lors d'un *Ctrl-POMME-Reset.* Comme cette opération détruit AUTOMATIQUEMENT le contenu du RAMdisk, j'ai prévu des protections :

- un *BRUN* accidentel sera sans conséquence puisqu'il forcera l'état *démarrage à chaud,*
- si vous avez un remords, après un *CALL 771,* il vous sera loisible de faire un *CALL 768* qui rétablira la situation standard (chaud).

J'espère que les commentaires du source vous permettront de comprendre le fonctionnement de ***FREEZER*** qui fait appel à plusieurs instructions spécifiques au 65816. Je laisse aux adeptes de *ProCODE* le soin de définir les macro-instructions remplaçant les opcodes que leur outil ignore.

Yvan KOENIG. (16 novembre 1987)



```
1            XC                ; 65c02
2            XC                ; 65816
3    ******************************************
4    * FREEZER.S        Yvan KOENIG   16-11-87 * 18/11/87
5    *----------------------------------------*
6    *              BLOAD FREEZER               *
7    *       Call 768+3  puis Ctrl POMME Reset  *
8    *                force un Reboot à froid   *
9    *          qui prend en considération les  *
10   *          indications du tableau de bord  *
11   *         sans avoir à éteindre / allumer  *
12   *                Si vous avez un remords   *
13   *          Call 768 exécuté avant le RESET *
14   *             Réinstalle le reboot à chaud *
15   ******************************************
16   NZ00      =      $00
17   NZ03      =      $03
18             ORG    $300
```

(suite page 18)

```
                        19
0300: A9 FF     20  CALL768  LDA   £$FF        ;pour forcer Démarrage CHAUD
0302: 2C        21           HEX   2C
0303: A9 00     22  CALL771  LDA   £0          ;pour forcer Démarrage FROID
0305: 8D 3A 03  23           STA   mode
0308: 18        24           CLC
0309: FB        25           XCE
030A: C2 30     26           REP   $30         ;Natif 16 bits
030C: 8B        27           PHB               ;Sauve banc source
030D: 0B        28           PHD               ;Sauve Page Directe
030E: F4 45 03  29           PEA   new00       ;Page zéro provisoire
0311: 2B        30           PLD               ;  en new00
                31
0312: F4 00 00  32           PEA   $0000       ;place pour réponse
0315: F4 00 00  33           PEA   $0000       ; .........
0318: F4 00 00  34           PEA   $0000       ;cherche adresse d'un ROMtool
031B: F4 02 01  35           PEA   $0102       ;     fonction 1 de l'outil 2
031E: A2 01 0B  36           LDX   £$0B01      ;Appel outil GetFuncPtr
0321: 22 00 00  37           DFB   $22,0,0,$E1 ; JSL $E10000
0324: E1
0325: 68        38           PLA
0326: 85 00     39           STA   NZ00        ;mot bas adresse MMBootInit-1
0328: 68        40           PLA
0329: 85 02     41           STA   NZ00+2      ;mot haut adresse MMBootInit-1
032B: A0 03 00  42           LDY   £0003       ;La 1ère instruction de
                43                             ; MMbootInit est  LDA XxYyZz
032E: B7 00     44           DFB   $B7,NZ00    ;LDA "NZ00§,Y ;récupère YyZz
0330: 85 04     45           STA   NZ03+1
0332: 88        46           DEY
0333: E2 30     47           SEP   $30         ; A,X,Y  8bits
0335: B7 00     48           DFB   $B7,NZ00    ;LDA "NZ00§,Y ;récupère Xx
0337: 85 03     49           STA   NZ03        ; désormais    NZ03:XxYyZz
0339: A9 FF     50           LDA   £$FF
                51  mode     =     *-1
033B: 88        52           DEY               ;-> Y=1
033C: 97 03     53           DFB   $97,NZ03    ;STA "NZ03§,Y ;Modifie l'octet
                54                             ;  haut du mot pointé par NZ03
033E: C2 30     55           REP   $30         ; A,X,Y  16 bits
0340: 2B        56           PLD               ;Retour page 0 initiale
0341: AB        57           PLB               ;Retour banc origine
0342: 38        58           SEC
0343: FB        59           XCE               ;Retour émulation
0344: 60        60           RTS               ;C'est vini!
0345: 00 00 00  61  new00    DS    3           ;En service ce sera NZ00
0348: 00 00 00  62  new03    DS    3           ;En service ce sera NZ03


--End assembly, 75 bytes, Errors: 0
```

Ne pas utiliser si vous avez démarré avec ProDOSQ 16 sur Apple IIgs ancienne ROM.

LANGAGE C | APPLE *IIGS*

Claude AUBRY

VITAMINES C ?

Je ne suis pas vraiment inquiet, mais mon toubib s'est mis à apprendre le langage **C** . Il y a deux ans, converti au Basic par mon dentiste (il avait piqué le logiciel de gestion de stock de mon pharmacien), j'ai passé l'hiver à soigner mon rhume avec des petites pilules pour le foie. Tout ça pour vous dire qu'il faut que, ce mois-ci, **C** soit vraiment simple, sinon de grandes catastrophes surviendront au moment de la dinde de Noël.

Au fait, qu'est-ce qui fait qu'un langage de programmation est simple à apprendre ?

Quatre éléments :

- le rapprochement du langage naturel,
- le fait qu'il soit proche d'un langage déjà connu,
- l'utilisation d'un vocabulaire réduit,
- l'emploi d'instructions très spécialisées.

Si vous avez lu les articles précédents, vous avez pu remarquer que **C** ne répond pas exactement au premier critère. Il faut bien reconnaître que, même avec la meilleure volonté du monde, un programme en **C** ne ressemblera jamais à du Molière. Toutefois, **C** est suffisamment "souple" pour s'adapter à différents styles de programmation. On reconnaît "l'accent" de l'ex-basicois, de l'ex-pascalien ou de l'ex-assembliste.

Et puis, **C** présente cette originalité paradoxale de posséder relativement peu de mots-clefs (en tout 28) et d'offrir des instructions très spécialisées (par exemple de gestion de fichiers ou d'entrée/sortie) performantes. Ceci est rendu possible par les différentes phases de la transformation de votre programme en code exécutable : un processeur, sorte de traitement de texte automatique, vient insérer des constantes ou des macro-instructions là où vous le désirez. C'est le rôle des instructions £define et £include.

Si vous avez écrit par exemple...

£define bip putch(7);

...chaque fois que dans le texte vous avez placé l'instruction bip, le préprocesseur remplace celle-ci par putch(7);

Ensuite, le compilateur traduit le texte que vous avez écrit en langage machine, mais en ne s'occupant que des 28 mots-clefs dont je parlais plus haut. Lorsqu'il rencontre un mot inconnu, il implante un appel à un sous-programme d'adresse indéfinie.

Le linker vient alors compléter la cérémonie ; il vérifie que le mot inconnu du compilateur correspond à un mot connu de la librairie (fichier.lib), il implante le sous-programme correspondant à la fonction et corrige l'adresse d'appel.

Voilà, vous avez tout compris : un langage très simple est un langage qui ne comporte que les fonctions algorithmiques de base (séquence d'action, test, itération), puis des macro-instructions et des bibliothèques de fonctions puissantes et adaptées à des cas généraux ou particuliers.

Une bibliothèque de commande de robot pourra très bien comporter les instructions LEVE__BRAS, TOURNE__a__GAUCHE, EMBRASSE__MAITRE etc. *(suite page 20)*

Ou encore une bibliothèque de fonction MAC-INTOSH fournira l'interface style «des souris et des hommes».
Ces commandes seront «comprises» par le linker de la même façon que si elles faisaient partie intégrante du langage de base.
Pour l'instant, et pour cette fois nous allons nous attacher à définir les instructions de base, les mots clefs du langage.

1. INSTRUCTIONS et BLOCS

La fonction algorithmique la plus simple est la séquence, ou suite d'instructions.En "C" le séparateur d'instruction est le point virgule.Plusieurs instructions peuvent être groupées de façon à constituer un bloc qui se comportera lui même comme une seule instruction. Les délimiteurs de bloc sont le é et è sur l'Apple français et le { et } sur l'Apple américain.

```
/*  Programme DROOPY */
main( )
é
int i;
for (i=0;i<=10;i++)
   printf("Hello ");
   printf("Happy people");
è
```

```
/* programme FRENCH DROOPY */
     main( )
     é
     int i;
          for (i=0;i<=10;i++)
          é
          printf("Salut a vous");
          printf("gens heureux");
          è
     è
```

DROOPY fera écrire 10 fois Hello suivit d'une fois Happy people
FRENCH DROOPY fera écrire 10 fois Salut à vous gens heureux.

2. LES INSTRUCTIONS DE TEST

Le deuxième niveau algorithmique est fourni par les instructions de test. Le test en langage "C" ressemble furieusement à son homologue basic IF sinon que le THEN est implicite : la syntaxe en est

```
if (expression) instruction_1;
 else  instruction_2;
```

La partie else est facultative. Si expression est différent de 0 alors instruction_1 est exécuté sinon c'est instruction_2 est exécuté. Bien entendu instruction peut être aussi un bloc d'instructions.

```
main( )
é
int x;
printf("Donnez un nombre :");
scanf("%d",&x);
 if (x <= 1000) printf("x est plus petit ou égal à 1000");
 else printf("x est plus grand que 1000");
è
```

En plus de l'instruction IF "C" possède une puissante instruction dédiée au choix multiple : l'instruction SWITCH ou aiguillage. La syntaxe de switch est :

```
switch (expression)
                é
                case constante_1 : instruction_1; break;
                case constante_2 : instruction_2; break;
                case constante_3 : instruction_3; break;
                case constante_4 :
                case constante_5 : instruction_4_5 :break;
                                ....etc.................
                default          : instruction_d; break;
                è
```

Dans le cas ou expression (qui doit être un entier) est égal à constante_1 instruction_1 (ou bloc d'instructions_1) est exécuté.
Deux constantes peuvent correspondre à une seule instruction (constante_4 et constante_5). Si expression ne correspond à aucune constante alors la clause default (facultative) est exécutée.

```
/* Programme MORSE (un peu simplifié) */
main( )
é
char c;
printf("Tapez une lettre");
scanf("%c",c);
switch(c)
        é
        case 'A':printf("._ ");break;
        case 'B':printf("_... ");break;
        case 'C':printf("_._. ");break;
        case 'D':printf("_.. ");break;
        /* Continuez si ça vous amuse ! */
        default :printf("Lettre inconnue");break;
        è
è
```

3. LES INSTRUCTIONS D'ITÉRATION

Les instructions d'itération que des gens sans scrupule vendent pour des boucles, servent à exécuter un certain nombre de fois un certain nombre d'instructions.

L'archétype des instructions d'itération est le FOR lui aussi proche cousin du FOR NEXT du BASIC. La syntaxe de «for» est très simple :

for (e1;e2;e3)

mais aussi très complète puisque e1 désigne la valeur de départ de la variable compteur, e2 désigne la valeur finale et e3 l'incrément.

(suite page 22)

Le plus souvent l'incrément est défini par les opérateurs «unaires» definis ci-dessous:

a = ++i incrémente i d'une unité et donne à a la valeur de i après l'incrémentation
a = i++ incrémente i d'une unité et donne à a la valeur de i avant l'incrémentation

Mais il peut aussi être défini par l'opérateur unaire -- (qui se comporte comme ++) ou par des opérateurs binaires d'assignation tels que += ou encore de décalage de bit >> et <<

i+=i attribue à i la valeur 2 i
i>>n décale i de n bits à droite
i<<n idem mais à gauche

Les décalages de bits sont là pour faire plaisir aux programmeurs en assembleur dont le sport favori consiste à faire «tourner» des octets comme d'autres font tourner les tables, ainsi qu'a ceux qui veulent effectuer des divisions ou des multiplications ou des divisions par des multiples de 2 rapides. En effet si i est egal à 1024 (en binaire 10000000000), i>>1 sera égal à 512 (en binaire 01000000000). Tenez, pour vous réchauffer les doigts, tapez le programme ci-dessous:

```
main()
é
int i;
 for (i=2;i<100;i+=i)  printf("%dçn",i);
 for (i=0x400;i>1;i=i>>1) printf("%xçn",i);
è
```

Les instructions WHILE et DO

A ce propos, ayant déjà parlé de while dans le dernier numéro une erreur sournoise s'était glissée (sans doute à cause du soleil) dans le texte il fallait lire en bas de la page 47 «tant qu'une condition est remplie» et non pas n'est pas remplie ! Je remercie les âmes vertueuses et laborieuses qui ont contribué à cet erratum.

```
while (expression)
     é
     instruction_1;
     instruction_2;
     è
```

```
do
     é
     instruction_1;
     instruction_2;
     è
while(expression);
```

Ces instructions font à peu près la même chose: répéter une instruction ou un bloc TANT QU'UNE CONDITION EST REMPLIE la seule différence étant que dans le cas du DO instruction_1 et instruction_2 sont exécutées au moins une fois puisque le test a lieu en fin d'exécution.

Un petit mot avant de vous laisser jouer avec votre compilateur et un petit programme spécialement écrit pour vous : l'instruction BREAK déjà rencontrée avec SWITCH permet de terminer brutalement l'exécution d'une itération et de se brancher sur l'instruction qui suit le è de fin de bloc.

L'instruction CONTINUE permet de se brancher au début de la boucle (sur le FOR le WHILE ou le DO). Ces instructions quoique pratiques ne participent pas à la structuration des programmes lorsque vous pouvez les éviter (et on le peut toujours) faites le.

```
/*
**+-----------------------------------------------------------------+
**!                          JEU DE NIM                             !
**!        une version simplifiée du célèbre jeu de NIM             !
**!  Programme C.Aubry pour TREMPLIN MICRO adaptation GS E.Schwarz  !
**+-----------------------------------------------------------------+
*/
/*  Un exemple de macro pour générer le ''s'' du pluriel */
#define pluriel(a) (a>1) ? 's' : ' '
/*  Variable globale */
int paquet = 0;
main()
é
char s[4];
char c;
printf("IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIçn");
printf("IIIIIIIIIIIIIIIII LE JEU DE NIM IIIIIIIIIIIIIIçn");
printf("IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIçnçnçn");
printf("çn***  Chacun à notre tour, Enlevons de 1 à 3 allumettes");
printf("çnd'un paquet d'allumettes représentées par les lettres I");
printf("çnCelui qui se retrouve avec une seule allumette a perdu !!!");
while (paquet < 5)
            é
            printf("çnçnNombre d'allumettes dans le jeu (minimum 5):");
            scanf("%d",&paquet);
            è
        printf("çnVoulez vous commencer ? (o ou n) ");
        scanf("%4s",s);
        affiche_jeu(paquet);
        if (s[0] == 'o' ùù s[0] == 'O') jeu_homme();
        while (paquet)
                    é
                    jeu_Apple();
                    jeu_homme();
                    è
è

/* à vous de jouer */

jeu_homme()
é
        int nombre = 0;
        if (paquet==1) perdu("vous");
        while (nombre < 1 ùù nombre > 3 ùù nombre > paquet)
```

(suite page 24)

```
            é
            printf ("çnCombien d'allumettes retirez vous (1..3) ? ");
            scanf("%d",&nombre);
            if (nombre < 1 ùù nombre > 3 ùù nombre > paquet)
                printf("Ne trichez pas !!!");
            è
          paquet -= nombre;
          affiche_jeu();
è

/*  à l'Apple de jouer */

jeu_Apple()
é
          int nombre, coup_perdant;
          if (paquet==1) perdu("moi");
          /* Quel est le prochain coup perdant ? */
          coup_perdant = (((paquet - 1) / 4) * 4 + 1);
          nombre = paquet - coup_perdant;
          if (nombre < 1 ùù nombre > 3) nombre = 1;
          paquet -= nombre;
        printf("çnJe prends %d
allumette%cçn",nombre,pluriel(nombre));
          affiche_jeu();
è

/* on montre le jeu sous forme de I */
affiche_jeu()
 é
  int i;
  printf("çn Il reste %d allumette%c -> ",paquet,pluriel(paquet));
  for (i=0;i<paquet;i++) printf("I");
  printf("çn");
 è

/* affiche le perdant si paquet = 1 et termine */
perdu(perdant)
char *perdant;
é
printf("çnUne seule allumette c'est perdu pour %s!!!çn",perdant);
exit(0);
è
```

Je viens de découvrir que ma psychanalyste elle-même apprend le "C" en ce moment. Quelle horreur, quelle abomination, que faire ?, où me tourner ?

Ho "C" !!!, You drive me crazy....

Joël DESNOUES

FORMATAGE 3,5

```
 5 BUF = 28672:D$ =  CHR$ (4): PRINT D$"PR£3": PRINT  CHR$ (17): PRINT
   D$"BLOAD FORMAT.BIN": PRINT D$"BLOAD BLOCO,A";BUF                    EEA9
10 HOME : INVERSE : PRINT  SPC( 39): PRINT : PRINT " Programme de form
   atage de Lecteur 3.5 ": PRINT  SPC( 39): NORMAL : VTAB 5: PRINT      0DEB
15 ENTREE = 24576: CALL ENTREE + 26                                     7C54
20 BUF = 28672:VOLUME =  PEEK (ENTREE + 22) +  PEEK (ENTREE + 23) * 25
   6                                                                    280C
25 VERT =  PEEK (37) + 2: VTAB VERT: FOR I = 0 TO 39: PRINT  CHR$ (95)
   ;: NEXT                                                             B487
30 PRINT : PRINT : PRINT " ¨Unité à formater..: ": VTAB VERT + 3: CAL
   L ENTREE + 29                                                       5E6B
32 GOSUB 200                                                           0942
35 VTAB VERT + 5: HTAB 13: PRINT "Nom ...: /": VTAB VERT + 5           0183
40 CALL ENTREE + 32                                                    267C
45 X =  PEEK (ENTREE + 1): VTAB VERT + 7: PRINT "Insérez une disquette
    dans l'unité ";X: PRINT : PRINT "et pressez une touche ..."        7293
50 HTAB 39: GET A$: VTAB 19: PRINT  CHR$ (11): VTAB 20: PRINT "Patienc
   e ..."                                                              959C
55 BUF = BUF + 512: FOR I = 0 TO 42:X =  PEEK (VOLUME + I): POKE BUF +
   I,X: NEXT                                                           ECB4
60 X = 0: FOR I = 43 TO 512: POKE BUF + I,X: NEXT                      DD0D
65 PRINT : PRINT "Formatage en cours ...": CALL ENTREE + 38            074D
70 CALL ENTREE + 47                                                    1E82
75 X =  PEEK (ENTREE + 3) + 2: POKE ENTREE + 3,X:Y =  PEEK (ENTREE + 4
   ) + 2: POKE ENTREE + 4,Y                                            ECCE
80 CALL ENTREE + 47: GOSUB 220                                         F300
85 NBL =  PEEK (ADR + 1) +  PEEK (ADR + 2) * 256:NBL = (NBL / 8) - 1:X
   = 255: FOR I = 1 TO NBL: POKE BUF + I,X: NEXT : POKE BUF,65: POKE E
   NTREE + 4,6                                                         5F2E
90 CALL ENTREE + 47                                                    1E82
95 VTAB VERT + 2: PRINT  CHR$ (11): PRINT "Terminé ..."                9C97
100 CALL ENTREE + 41                                                   127C
110 PRINT : VTAB 22: PRINT "(1) ENCORE (2) MENU DISQUETTE (3) FIN";: PO
   KE 1745,49: POKE 1756,50: POKE 1775,51: GET R$: PRINT               E926
120 ON  VAL (R$) GOTO 10,130,140                                       68ED
130 REM ? CHR$(4)"RUN MENU"
140 HOME : END                                                         8E51
200 ADR =  PEEK (ENTREE + 24) +  PEEK (ENTREE + 25) * 256:X = ADR -  IN
   T (ADR / 256) * 256:Y =  INT (ADR / 256): POKE ENTREE + 2,X: POKE E
   NTREE + 3,Y: CALL ENTREE + 35                                       DCD4
205 X = BUF -  INT (BUF / 256) * 256:Y =  INT (BUF / 256): POKE ENTREE
   + 2,X: POKE ENTREE + 3,Y: POKE ADR,0                                3FFA
210 RETURN                                                             63B1
220 X = 0: FOR I = 0 TO 511: POKE BUF + I,X: NEXT                      CCD5
230 X = 1:Y = 3: FOR I = 3 TO 5:X = X + 1:Y = Y + 1: IF Y = 6 THEN Y =
   0                                                                   FEB7
240 POKE ENTREE + 4,I: POKE BUF,X: POKE BUF + 2,Y: CALL ENTREE + 47: NE
   XT                                                                  A308
250 RETURN                                                             63B1
```

(Routines LM au verso)

FORMATAGE 3,5

```
FORMAT/FORMAT.BIN,A$6000,L$0200

6000: 00 00 00 70 00 00 00 00 00 00 00 00 00 00 00 00   7270
6010: 00 00 00 00 00 00 D5 61 FD 61 4C 81 60 4C 10 61   717E
6020: 4C 3D 61 4C 7D 60 4C 70 60 4C 76 60 18 90 01 38   7232
6030: A9 01 69 00 A2 03 8E 00 60 8D 3F 60 20 00 00 03   AAF5
6040: 00 60 8D 07 60 90 28 C9 27 F0 24 A2 03 2C A2 00   FC83
6050: 20 58 FC 20 64 60 A9 A4 20 F6 FD AD 07 60 20 DA   BBC6
6060: FD 4C 2A D4 BD 9F 61 F0 06 20 F6 FD E8 D0 F5 60   181A
6070: A9 03 A2 01 D0 C0 A9 04 8D 04 60 D0 B7 A9 00 F0   CC9D
6080: B3 A9 00 85 06 8D 07 60 8D 04 60 A2 0D 9D 08 60   1A80
6090: CA 10 FA A2 07 A9 C7 85 07 A0 07 B1 06 D9 97 61   0AA8
60A0: F0 07 C6 07 CA D0 F2 F0 A5 88 88 10 EE C6 06 A0   EA5F
60B0: 00 B1 06 18 69 03 8D 3D 60 A5 07 8D 3E 60 A2 0D   BFEB
60C0: A0 00 BD 32 BF 29 0F C9 0B D0 09 BD 32 BF 29 F0   F0FA
60D0: 99 08 60 C8 CA 10 EB E8 8E D4 61 BD 08 60 F0 2F   697D
60E0: DA 18 0A 48 A9 B1 69 00 8D C6 61 68 4A 4A 4A 4A   964B
60F0: 4A 18 69 B0 8D BC 61 AD D4 61 69 B1 8D D4 61 20   B203
6100: 8E FD A0 61 A9 B7 20 3A DB 20 8E FD FA 10 C8 6U   5BFE
6110: 20 E4 FB 20 2A 61 20 31 61 91 28 29 0F AA 8E 01   C886
6120: 60 F0 ED CA BD 08 60 F0 E7 60 A0 17 A9 FF 91 28   F97B
6130: 60 2C 10 C0 AD 00 C0 10 FB 2C 10 C0 60 20 2A 61   DDDB
6140: A2 00 8E 7B 05 20 31 61 C9 FF F0 2D C9 8D F0 37   5FC4
6150: C9 DB B0 F1 C9 C1 B0 12 C9 B0 B0 06 C9 AE F0 06   C72D
6160: D0 E3 C9 BA B0 DF E0 00 F0 DB E0 0F F0 D7 91 28   8FDF
6170: 29 7F 9D DA 61 C8 E8 10 09 E0 00 F0 C8 20 92 61   2DF4
6180: CA 88 20 2C 61 D0 BE E0 00 F0 BA 8A 18 69 F0 8D   9F9F
6190: D9 61 A9 A0 91 28 60 FF 20 FF 00 FF 03 FF 00 CE   6289
61A0: CF A0 D0 D2 CF D4 CF C3 CF CC A0 C3 CF CE D6 C5   FC7C
61B0: D2 D4 C5 D2 AE A0 00 D3 EC EF F4 A0 00 A0 CC E5   431E
61C0: E3 F4 E5 F5 F2 A0 00 A0 D5 EE E9 F4 FB A0 AE AE   9F7A
61D0: AE AD BE A0 00 00 00 03 00 00 00 00 00 00 00 00   09BC
61E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   0000
61F0: 00 00 00 00 00 00 00 C3 27 0D 00 00 06 00 40 06   6C43

FORMAT/BLOC0,A$7000,L$0200

7000: 01 38 B0 03 4C 32 A1 86 43 C9 03 08 8A 29 70 4A   DC15
7010: 4A 4A 4A 09 C0 85 49 A0 FF 84 48 28 C8 B1 48 D0   E599
7020: 3A B0 0E A9 03 8D 00 08 E6 3D A5 49 48 A9 5B 48   0CDE
7030: 60 85 40 85 48 A0 63 B1 48 99 94 09 C8 C0 EB D0   1967
7040: F6 A2 06 BC 1D 09 BD 24 09 99 F2 09 BD 2B 09 9D   668C
7050: 7F 0A CA 10 EE A9 09 85 49 A9 86 A0 00 C9 F9 B0   8712
7060: 2F 85 48 84 60 84 4A 84 4C 84 4E 84 47 C8 84 42   D8A9
7070: C8 84 46 A9 0C 85 61 85 4B 20 12 09 B0 68 E6 61   9897
7080: E6 61 E6 46 A5 46 C9 06 90 EF AD 00 0C 0D 01 0C   377F
7090: D0 6D A9 04 D0 02 A5 4A 18 6D 23 0C A8 90 0D E6   D48A
70A0: 4B A5 4B 4A B0 06 C9 0A F0 55 A0 04 84 4A AD 02   9574
70B0: 09 29 0F A8 B1 4A D9 02 09 D0 DB 88 10 F6 29 F0   8D1A
70C0: C9 20 D0 3B A0 10 B1 4A C9 FF D0 33 C8 B1 4A 85   70B2
70D0: 46 C8 B1 4A 85 47 A9 00 85 4A A0 1E 84 4B 84 61   6CBF
70E0: C8 84 4D 20 12 09 B0 17 E6 61 E6 61 A4 4E E6 4E   B34F
70F0: B1 4A 85 46 B1 4C 85 47 11 4A D0 E7 4C 00 20 4C   8859
7100: 3F 09 26 50 52 4F 44 4F 53 20 20 20 20 20 20 20   1D25
7110: 20 20 A5 60 85 44 A5 61 85 45 6C 48 00 08 1E 24   6DDC
7120: 3F 45 47 76 F4 D7 D1 B6 4B B4 AC A6 2B 18 60 4C   F7D3
7130: BC 09 A9 9F 48 A9 FF 48 A9 01 A2 00 4C 79 F4 20   2E6A
7140: 58 FC A0 1C B9 50 09 99 AE 05 88 10 F7 4C 4D 09   829F
7150: AA AA AA A0 D5 CE C1 C2 CC C5 A0 D4 CF A0 CC CF   D6D3
7160: C1 C4 A0 D0 D2 CF C4 CF D3 A0 AA AA AA A5 53 29   80BB
7170: 03 2A 05 2B AA BD 80 C0 A9 2C A2 11 CA D0 FD E9   730C
7180: 01 D0 F7 A6 2B 60 A5 46 29 07 C9 04 29 03 08 0A   721F
7190: 28 2A 85 3D A5 47 4A A5 46 6A 4A 4A 85 41 0A 85   4588
71A0: 51 A5 45 85 27 A6 2B BD 89 C0 20 BC 09 E6 27 E6   3996
71B0: 3D E6 3D B0 03 20 BC 09 BC 88 C0 60 A5 40 0A 85   EED0
71C0: 53 A9 00 85 54 A5 53 85 50 38 E5 51 F0 14 B0 04   B3C8
71D0: E6 53 90 02 C6 53 38 20 6D 09 A5 50 18 20 6F 09   DE57
71E0: D0 E3 A0 7F 84 52 08 28 38 C6 52 F0 CE 18 08 88   7E8E
71F0: F0 F5 BD 8C C0 10 FB 00 00 00 00 00 00 00 00 00   21F9
```

```
 1 REM ******************
 2 REM * CREATION BLOC0 *
 3 REM ******************
 4 REM
 5 BUF = 28672:D$ =  CHR$ (4):
   PRINT CHR$ (17): PRINT D$"BL
   OAD FORMAT.BIN"                    7B1A
10 HOME : INVERSE : PRINT  SPC(
   39): PRINT : PRINT " ¯¯Progr
   amme de Création d'un Bloc 0
   ¯¯": PRINT  SPC( 39): NORMA
   L : VTAB 5: PRINT                  A38B
15 ENTREE = 24576: CALL ENTREE
   + 26: REM Initialisation           7C54
25 VERT =  PEEK (37) + 2: VTAB
   VERT: FOR I = 0 TO 39: PRINT
   CHR$ (95);: NEXT                   B487
30 PRINT : PRINT : PRINT " ¯¯Un
   ité ¯Source ....: ": VTAB VE
   RT + 3: CALL ENTREE + 29: RE
   M Choix du Drive                   7D18
35 CALL ENTREE + 44: REM Lectur
   e du Bloc 0                        1C7F
40 PRINT D$"BSAVE BLOC0,A";BUF;
   ",L512"                            1B5B
45 VTAB VERT + 2: PRINT  CHR$ (
   11): PRINT "Terminé ..."           9C97
50 END                                0180
```

LIT.JONG

UTILISATION : Uniquement par BRUN LIT.JONG. On indique alors le chemin et le titre choisis.
Exemples :

- TOT/TITI pour lire le fichier TITI, du sous-catalogue TOT (cela suppose que l'on a pour PREFIXE celui de la disquette concernée).
- /JONGLEUR/TOT/TITI pour lire le fichier TITI du sous-catalogue TOT, sur une disquette dont le PREFIXE est JONGLEUR.

Le Jongleur de mots, traitement de texte paru en anglais sous le titre *World Juggler* (chez Quark Inc.), stocke astucieusement les voyelles surmontées d'un tréma ou d'un accent circonflexe*. En effet, alors que tous les autres caractères (codes divers exceptés) sont codés en ASCII, les â ê î ô û sont traduits par 3, 4, 5, 6 ou 7, et les ä ë ï ö ü par 8, 9, A, B ou C.

Cette petite routine charge un fichier (TXT) à partir de l'adresse $1000, puis affiche les lignes du texte en respectant sa mise en page (retours ligne). Les voyelles seront donc affichées, mais sans accent circonflexe ou tréma. Par contre, dans le même temps, le fichier est réécrit à partir de l'adresse $900, accents litigieux compris, précédant les caractères concernés. On pourra lire l'adresse de fin de fichier en $8-9.

LIT-JONG aidera les utilisateurs de ***Jongleur de mots*** à vérifier le contenu d'un texte sans avoir à charger le logiciel. L'affichage se fait par page de 22 lignes (n'importe quelle touche) ou ligne par ligne (barre d'espacement).

* ***Le Jongleur de mots*** *utilise un écran graphique de 80 colonnes, ce qui lui permet d'afficher tous les caractères accentués. C'est un excellent traitement de texte.*

SAISIE DU TITRE DU FICHIER À LIRE

```
00/0801 : 20 58 FC    JSR  FC58
00/0804 : A0 08       LDY  £08
00/0806 : A9 C9       LDA  £C9
00/0808 : 20 3A DB    JSR  DB3A
00/080B : 20 6F FD    JSR  FD6F
00/080E : 8A          TXA
00/080F : F0 33       BEQ  0844
00/0811 : A8          TAY
```

HOME efface l'écran.

Adresse de la chaîne "BLOAD".

Strout affiche la chaîne pointée par Y/A.
Une entrée de GETLIN (ponctuation acceptée).
Le nombre de caractères est dans X.
S'il est nul, terminé.
A dans Y (=X).

TITRE EMPILÉ

```
00/0812 : BD FF 01    LDA  01FF,X
00/0815 : 48          PHA
00/0816 : CA          DEX
00/0817 : D0 F9       BNE  0812
```

Lecture du texte saisi (il est empilé à l'envers).
A la sortie, X = 0.

(suite page 28)

BLOAD DANS LE BUFFER

```
00/0819 : BD C9 08    LDA 08C9,X
00/081C : F0 06       BEQ 0824
00/081E : 9D 00 02    STA 0200,X
00/0821 : E8          INX
00/0822 : D0 F5       BNE 0819
00/0824 : C8          INY
```

On installe BLOAD dans le buffer.
Quand l'accumulateur contient 0, sortie.
X est alors prêt pour le prochain caractère.

Y est ajusté pour autoriser l'opération suivante.

TITRE DANS LE BUFFER

```
00/0825 : 88          DEY
00/0826 : F0 07       BEQ 082F
00/0828 : 68          PLA
00/0829 : 9D 00 02    STA 0200,X
00/082C : E8          INX
00/082D : D0 F6       BNE 0825
```

Y = d'abord la longueur du texte saisi (X).
Quand Y = 0, sortie.

Un caractère dépilé est écrit dans le buffer.

X est incrémenté... pour écriture.
L'opération continue.

TYPE ET ADRESSE DANS LE BUFFER

```
00/082F : A0 07       LDY £07
00/0831 : B9 C9 08    LDA 08C9,Y
00/0834 : 9D 00 02    STA 0200,X
00/0837 : C9 8D       CMP £8D
00/0839 : F0 04       BEQ 083F
00/083B : E8          INX
00/083C : C8          INY
00/083D : D0 F2       BNE 0831
00/083F : 20 03 BE    JSR BE03
00/0842 : 90 01       BCC 0845
00/0844 : 60          RTS
```

On doit lire et écrire ",TTXT,A$1000" et 8D.
On aurait pu donner Y = 0 et l'adresse vraie du texte : $8D0.

Quand on aura lu et écrit $8D, terminé !

Les deux compteurs sont incrémentés.

La boucle continue.
MLI.
Si la retenue est à 0, pas d'erreur : on saute.
FIN.

LONGUEUR DU FICHIER

```
00/0845 : AD C8 BE    LDA BEC8
00/0848 : 85 06       STA 06
00/084A : AD C9 BE    LDA BEC9
00/084D : 18          CLC
00/084E : 69 10       ADC £10
00/0850 : 85 07       STA 07
```

A la longueur, lue en $BEC8-BEC9,
on ajoute $10 (partie haute),
adresse de début de stockage.

FIN DE FICHIER

```
00/0852 : A9 00       LDA £00
00/0854 : A8          TAY
00/0855 : 91 06       STA (06),Y
00/0857 : EA          NOP
```

A = 0 et Y = 0.

On met un 0 en fin de fichier.

POINTEURS DIVERS

```
00/0858 : 85 06       STA 06
00/085A : 85 08       STA 08
00/085C : A9 09       LDA £09
00/085E : 85 09       STA 09
00/0860 : A9 10       LDA £10
00/0862 : 85 07       STA 07
```

$6-7 : adresse du fichier (chargement).
$8-9 : adresse du fichier réécrit.

NOMBRE DE LIGNES PAR PAGE

```
00/0864 : A9 15       LDA £15
00/0866 : 85 18       STA 18
```

Nombre de lignes limité à $15 (0-21).

DÉCRYPTAGE

```
00/0868 : B1 06     LDA (06),Y
00/086A : F0 D8     BEQ 0844
00/086C : C9 20     CMP £20
00/086E : B0 20     BCS 0890
00/0870 : C9 03     CMP £03
00/0872 : 90 34     BCC 08A8
00/0874 : C9 0D     CMP £0D
00/0876 : F0 18     BEQ 0890
00/0878 : B0 2E     BCS 08A8
```

Y vaut 0 (voir page 28).
Quand l'accumulateur contiendra 0 : terminé.

Si supérieur ou égal à l'espace, bon.

Si inférieur à 3, aucun affichage.

Si c'est RETURN, traitement de faveur.

Supérieur à $D, aucun affichage.

âêîôû et äëïöü

```
00/087A : AA        TAX
00/087B : C9 08     CMP £08
00/087D : B0 04     BCS 0883
00/087F : A9 DE     LDA £DE
00/0881 : D0 02     BNE 0885
00/0883 : A9 FE     LDA £FE
00/0885 : 91 08     STA (08),Y
```

A dans X (â = 3 - ä = 8).

Egal ou supérieur à 8, c'est le tréma : saut.

Accent circonflexe dans A et saut.

Tréma dans A.
Ecriture dans le nouveau fichier.

TRANSFORMATION DU CARACTÈRE ACCENTUÉ

```
00/0887 : BD DA 08  LDA 08DA,X
00/088A : E6 08     INC 08
00/088C : D0 02     BNE 0890
00/088E : E6 09     INC 09
00/0890 : 09 80     ORA £80
```

On lit le bon caractère.

Compteur d'écriture incrémenté (pour tenir compte de l'accent).

Bit 7 à 1.

VÉRIFICATION FIN DE LIGNE

```
00/0892 : C9 A0     CMP £A0
00/0894 : D0 07     BNE 089D
00/0896 : AE 7B 05  LDX 057B
00/0899 : E0 4F     CPX £4F
00/089B : F0 03     BEQ 08A0
```

Quand il y a un espace en fin de ligne, le return qui suit provoque un double saut de ligne. Ce contrôle évite ce désagrément (on n'affiche pas l'espace).

AFFICHAGE DU CARACTÈRE

```
00/089D : 20 ED FD  JSR FDED
00/08A0 : 91 08     STA (08),Y
```

Affichage et écriture dans le nouveau fichier.

COMPTEUR D'ÉCRITURE

```
00/08A2 : E6 08     INC 08
00/08A4 : D0 02     BNE 08A8
00/08A6 : E6 09     INC 09
```

On incrémentera la partie haute si $FF + 1 = 0.

COMPTEUR DE LECTURE

```
00/08A8 : E6 06     INC 06
00/08AA: D0 02      BNE 08AE
00/08AC : E6 07     INC 07
```

NOMBRE DE LIGNES

```
00/08AE : C9 8D     CMP £8D
00/08B0 : D0 B6     BNE 0868
00/08B2 : C6 18     DEC 18
00/08B4 : 10 B2     BPL 0868
```

Est-ce RETURN ? (***Le Jongleur de mots*** en met un à la fin de chaque ligne). Non : suite.

Si oui, on décrémente le compteur jusqu'à 0 inclus.

(suite page 30)

ATTENTE

```
00/08B6 :  2C  10  C0      BIT  C010
00/08B9 :  AD  00  C0      LDA  C000
00/08BC :  10  FB          BPL  08B9
00/08BE :  2C  10  C0      BIT  C010
00/08C1 :  C9  A0          CMP  £A0
00/08C3 :  D0  9F          BNE  0864
00/08C5 :  E6  18          INC  18
00/08C7 :  80  9F          BRA  0868
```

Attente d'une touche pour continuer.

Si ce n'est pas l'espace, suite normale, page par page.

Si c'est l'espace, le compteur est incrémenté pour une autre ligne.

TEXTES et TABLE DES VOYELLES ACCENTUÉES

```
*8C9.8E6
00/08C9 :  C2 CC CF C1 C4 A0 00 - BLOAD.
00/08D0 :  AC D4 D4 D8 D4 AC C1 A4 B1 B0 B0 B0 8D E1 E5 E9  -,TTXT,A$1000.aei
00/08E0 :  EF F5 E1 E5 E9 EF F5 - ouaeiou
```

BSAVE LIT.JONG, A$801, L$E6

UTILISATION AVEC L'AMPERSAND

Tel qu'il se présente, LIT.JONG exige un BRUN lors de chaque lecture de fichier. Les quelques lignes ci-après créent un programme EXEC qui autorise de multi-lectures par l'intermédiaire de l'Ampersand.

Si vous regardez à l'adresse $300, vous y trouverez :

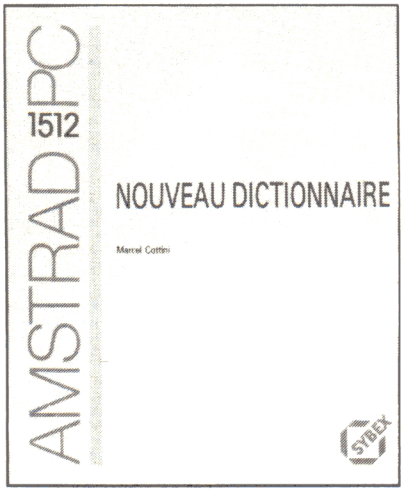
```
300 :  20  01  08      JSR  $0801  =  Adresse du programme.
303 :  4C  3C  D4      JMP  $D43C  =  CMDLOOP, boucle principale de l'interpréteur.
```

Bien entendu, personne ne vous interdit de réécrire LIT.JONG en y incorporant le JMP à $D43C et pourquoi pas ? — l'écriture de l'adresse $801 dans les paramètres de l'Ampersand ($3F6-3F7). Il ne sera plus nécessaire, après cela, de passer par un EXEC... ou par une succession de BRUN.

- Notez que l'affichage doit être en 80 colonnes.

```
10  D$ = CHR$ (4): PRINT : PRINT D$"OPEN STARTUP":
    PRINT D$"WRITE STARTUP"
20  PRINT "POKE 1013,76: POKE 1014,0: POKE 1015,3:
    POKE 768,32: POKE 769,01: POKE 770,8: POKE 771,76:
    POKE 772,60: POKE 773,212"
30  PRINT "PR£3": PRINT "HOME": PRINT "CATALOG"
40  PRINT "BRUN LIT.JONG"
50  PRINT D$"CLOSE STARTUP"                          ■
```

Votre bibliothèque INFORMATIQUE



• NOUVEAU DICTIONNAIRE DE L'AMSTRAD PC 1512 (par Marcel COTTINI)

Basé sur le standard des PC, avec un must au niveau du système d'exploitation, en l'occurrence le DOS Plus, le PC 1512 est une machine conviviale dans bien des domaines. Avec un rapport qualité/prix des plus compétitifs, ce type de micro-ordinateur devrait trouver acquéreur à sa mesure, qu'il soit "hobbyiste" ou homme d'affaires (PME/PMI).

Le ***Nouveau Dictionnaire Amstrad-PC 1512*** est un ouvrage très complet sur le PC 1512, regroupant en un seul volume toutes les informations essentielles pour maîtriser correctement une telle machine. Conçu sur le principe rédactionnel d'un dictionnaire, *LE NOUVEAU DICTIONNAIRE* est un ouvrage de référence de haut niveau destiné aux programmeurs hobbyistes sur PC, et particulièrement sur l'Amstrad PC 1512. Il développe en trois chapitres, un dictionnaire relatif à tout ce qui touche l'unité centrale et les systèmes d'exploitation MS-DOS et DOS Plus. Vous apprendrez dans un premier temps à vous familiariser avec l'environnement MS-DOS et DOS Plus, en donnant chaque fois la syntaxe des commande internes et externes. Un grand nombre d'exemples viennent conforter la théorie.

• **CHAPITRE 1 :** Ce chapitre est consacré à tout ce qui se rapporte en général au PC 1512, décrivant plus particulièrement les termes et les mots techniques, comme par exemple le microprocesseur, les contrôleurs d'interfaces, le Cluster, le Directory, etc. C'est en somme un dictionnaire alphabétique des mots usuels et particuliers, directement liés à la machine.

• **CHAPITRE 2 :** Il traite plus particulièrement le système d'exploitation MS-DOS, version 3.20 de MicroSoft. Toutes les commandes sont passées en revue, qu'elles soient externes ou internes, avec ou sans paramètres optionnels. Pour illustrer les cas difficiles, plusieurs exemples appuient la théorie.

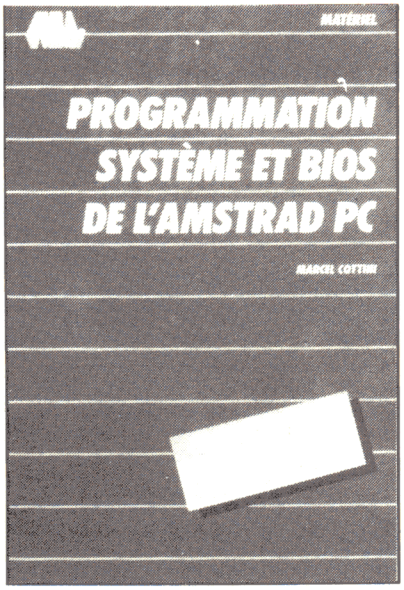
• **CHAPITRE 3 :** Il traite le système d'exploitation DOS Plus, disposition particulière du PC 1512 pour gérer les fichiers CP/M. La même structure descriptive que celle utilisée pour le traitement de MS-DOS sera adoptée.

(572 pages — Prix 198 F).

***Editions du SYBEX**, 6-8 impasse du Curé, 75018 PARIS.*

• PROGRAMMATION SYSTÈME ET BIOS DE L'AMSTRAD PC (par Marcel COTTINI)

Cet ouvrage regroupe en un seul volume toutes les informations essentielles pour pouvoir maîtriser correctement l'Amstrad PC 1512. Il s'adresse au programmeur sur Amstrad PC ou à l'utilisateur averti désirant mieux connaître sa machine. Programmation Système et BIOS de l'Amstrad PC vous permettra de tout savoir (ou presque tout) aussi bien sur le matériel que sur le BIOS et les interruptions du PC 1512. Après une description générale de la machine, les différents composants sont passés en revue, mettant l'accent sur la programmation des différents registres de chaque interruption du BIOS.



L'unité centrale, le contrôleur d'interruption, l'interface programmable de périphériques, le contrôleur vidéo, le port série, le contrôleur du lecteur de disquettes, et bon nombre d'autres organes essentiels sont détaillés un à un. Chaque structure est traitée dans le but de mettre à la disposition du lecteur tous les outils d'aide à la programmation. Cet ouvrage est indispensable pour toute personne voulant aller au cœur de l'Amstrad PC, vous donnant toutes les informations pour mieux tirer parti de votre machine. Un glossaire dresse une liste non exhaustive du vocabulaire informatique spécifique à l'environnement Amstrad PC afin d'attirer l'attention du lecteur sur des cas d'interprétation tendancieuses.

Pour les plus exigeants, des annexes révèlent des détails croustillants et des astuces pour satisfaire les plus "branchés" de la micro-informatique. Un ouvrage à lire et à relire !

(245 pages — Prix 249 F)

***Editions du PSI**, 5 place du Colonel Fabien, 75010 PARIS*

• ALGORITHMES ET STRUCTURES DE DONNÉES (par Nicklaus WIRTH)

"L'ordinateur a été inventé pour exécuter plus facilement des calculs très longs et très complexes ; toutefois, dans la majorité des applications de l'informatique, c'est la possibilité de stocker et de manipuler un grand nombre de données qui est primordiale, tandis que la capacité à calculer, à exécuter des opérations arithmétiques est très secondaire."

Ces quelques lignes sont les premières du chapitre 1 de l'excellent ouvrage de N. WIRTH et on peut dire qu'elles donnent parfaitement le ton de ce livre sérieux, documenté, riche en exemples utiles et bien commentés.

Attention ! ce n'est pas un roman, mais plutôt un cours écrit par un bon pédagogue. Le langage utilisé est le Modula-2, fruit de dix années de Pascal dont il garde les concepts de base tout en y incorporant quelques améliorations et extensions. A défaut de pratiquer le Modula-2, une bonne connaissance du Pascal est indispensable pour tirer parti des riches enseignements de ce livre (reliure solide permettant un usage intensif... un bon point pour l'éditeur !). **Nestor.**

***Editions EYROLLES**, 61, boulevard St-Germain 75240 PARIS Cédex 05 (336 pages — Prix 290 F)*

Clément RENARD

SBC.ADC 65816 (APPLE *IIGS*)

Soustraction élémentaire sur 16 bits, suivie d'une addition qui permet de retrouver la valeur utilisée au départ.

SOUSTRACTION

```
00/0300 : 18          CLC
00/0301 : FB          XCE
00/0302 : C2 30       REP  #$30
00/0304 : A9 FF FF    LDA  #$FFFF
00/0307 : 38          SEC
00/0308 : E9 00 D0    SBC  #$D000
00/030B : 85 06       STA  $06
00/030D : 38          SEC
00/030E : FB          XCE
00/030F : 60          RTS
```

Mode natif pur.

Le nombre est $FFFF (16 bits)*.
Retenue à 1 pour soustraire.
Le nombre à déduire est $D000 (16 bits).
Le résultat est stocké page 0, aux adresses $6... et $7.

Retour au mode émulation.

ADDITION

```
00/0310 : 18          CLC
00/0311 : FB          XCE
00/0312 : C2 30       REP  #$30
00/0314 : A9 00 D0    LDA  #$D000
00/0317 : 18          CLC
00/0318 : 65 06       ADC  $06
00/031A : 85 06       STA  $06
00/031C : 38          SEC
00/031D : FB          XCE
00/031E : 60          RTS
```

Mode natif pur.

On a toujours $D000.
Annulation de la retenue pour additionner.
On additionne le contenu des adresses $6... et $7.
Et on écrase l'ancienne valeur avec la nouvelle.

Retour.

* **Remarque :** On aurait pu lire le nombre de base dans deux mémoires de la page zéro par A5 08, par exemple (nombre stocké en $8-9, partie basse dans $8 et partie haute dans $9).

Vous pouvez faire un essai en remplaçant : 304 : A9 FF FF par 304 : A5 08 EA (EA = NOP, sans effet) ; placez ensuite votre nombre en $8-9. N'oubliez jamais, lorsque vous programmez en mode natif sur votre Apple *IIGS*, que les instructions prennent ainsi (pas toutes !) une autre dimension.

Vérifiez cette double routine en direct, à partir du moniteur. Pour obtenir un listage éventuel correct, ne pas oublier de taper :

0=m : 0=x **RETURN**

```
*300G                 | *310G
*6.7                  | *6.7
00/0006:FF 2F-./      | 00/0006:FF FF-..
```

NESTOR

DIFFÉRENCES

On me demande souvent : "Pourquoi programmer en langage machine ?". Voici la réponse. Sur un Apple *IIc*, elle crève les yeux !

```
300 : 20 84 FE    JSR  $FE84     SETNORM : affichage en mode normal.
303 : 20 39 FB    JSR  $FB39     SETTXT : mode texte.
306 : 20 58 FC    JSR  $FC58     HOME pour effacer l'écran.

309 : A9 03       LDA  £$03   ]  03 dans CV pour VTAB 4.
30B : 85 25       STA  $25    ]
30D : 20 22 FC    JSR  $FC22     VTAB déplace le curseur en CV.
310 : 20 32 03    JSR  $0332     Sous-programme pour tracer une ligne.
313 : 20 62 FC    JSR  $FC62     CR envoie un retour chariot (ou LF, $FC66).
316 : A4 25       LDY  $25    ]  A-t-on atteint la limite
318 : C0 11       CPY  £$11   ]  (ici fixée à $11/17 = VTAB 18) ?
31A : F0 12       BEQ  $032E     Si oui, saut.

31C : A2 04       LDX  £$04   ]  CH installé.
31E : 86 24       STX  $24    ]
320 : A9 AA       LDA  £$AA   ]  Affichage du caractère (on pourrait aller le lire dans
322 : 20 ED FD    JSR  $FDED  ]  une adresse de la page 0 par A5 06, par exemple).
325 : A2 22       LDX  £$22   ]  CH fin de la ligne ($22 = 34 soit HTAB 35).
327 : 86 24       STX  $24    ]
329 : 20 ED FD    JSR  $FDED     Affichage.
32C : 80 E5       BRA  $0313     Et on continue.

32E : 20 32 03    JSR  $0332     Vers le sous-programme pour le bas du cadre.
331 : 60          RTS            Retour au Basic.

332 : A0 04       LDY  £$04   ]  CH installé.
334 : 84 24       STY  $24    ]
336 : A9 AA       LDA  £$AA      Caractère désiré dans A.
338 : 20 ED FD    JSR  $FDED     COUT affiche.
33B : C8          INY            Y = Y + 1.
33C : C0 23       CPY  £$23   ]  Quand Y atteint la limite prévue, on arrête.
33E : D0 F8       BNE  $0338  ]
340 : 60          RTS            Retour.
```

■

BRUN EX3

On peut faire mieux, mais c'est déjà très bien. Quant à la longueur, il suffit de consulter le catalogue ProDOS pour constater que cet **EX3** est plus court que les deux exemples en Basic. *(suite page 34)*

RUN EX1

Cette routine n'est vraiment pas optimisée. Bel exemple (voulu) de lenteur !

```
10  TEXT : NORMAL : HOME
20  FOR I = 4 TO 18
30  FOR J = 5 TO 35
40  IF (I > 4 AND I < 18) AND (J > 5 AND J < 35) THEN 60
50  VTAB I: HTAB J: PRINT "* "
60  NEXT : NEXT
```

RUN EX2

C'est déjà mieux. Sur un GS, c'est même suffisant.

```
10  TEXT : NORMAL : HOME
20  FOR I = 1 TO 31:A$ = A$ + "* ": NEXT
30  VTAB 4: HTAB 5: PRINT A$
40  FOR I = 5 TO 17: HTAB 5: PRINT "* ";: HTAB 35: PRINT "* ": NEXT
50  HTAB 5: PRINT A$
```

LIGNE

Ligne horizontale instantanée

```
10  TEXT : HOME : D$ = CHR$ (4): PRINT CHR$ (21): HOME : GOSUB 60
15  VTAB 20: INPUT "CARAC";C$:C = ASC (C$)
20  VTAB 22: INPUT "Vert,Hdébut,Hfin  ";V,HD,HF
25  CALL 768,C,V,HD,HF
30  CALL - 198: POKE 49168,0: WAIT 49152,128: POKE 49168,0
35  VTAB 24: PRINT "(E)NCORE  (A)PPLESOFT  (M)ENU  ";: GET R$: VTAB 22: PRINT
40  IF R$ = "E" OR R$ = "e" THEN 15
45  IF R$ = "M" OR R$ = "m" THEN PRINT D$"RUN MENU"
50  IF R$ = "A" OR R$ = "a" THEN HOME : END
55  GOTO 35
60  FOR I = 768 TO 802: READ R:POKEI,R: NEXT : RETURN
65  DATA 32,245,230,138,9,128,72,32,245,230,202,138,32,71,248,32,245,230,202,218,32,245,
    230,134,6,122,104,145,38,200,196,6,144,249,96
```

```
300 :  20  F5  E6    JSR  $E6F5
303 :  8A            TXA
304 :  09  80        ORA  £$80
306 :  48            PHA
307 :  20  F5  E6    JSR  $E6F5
30A :  CA            DEX
30B :  8A            TXA
30C :  20  47  F8    JSR  $F847
30F :  20  F5  E6    JSR  $E6F5
312 :  CA            DEX

313 :  DA            PHX
314 :  20  F5  E6    JSR  $E6F5
317 :  86  06        STX  $06
319 :  7A            PLY
31A :  68            PLA
31B :  91  26        STA  ($26),Y
31D :  C8            INY
31E :  C4  06        CPY  $06
320 :  90  F9        BCC  $031B
322 :  60            RTS
```

LOCNBR

OBJET Saisir des nombres — et rien que des nombres — à partir d'un point précis de l'écran. On efface avec la flèche arrière. Fin de saisie avec RETURN. La longueur du nombre est limitée à **N** chiffres. Sa valeur est contenue dans une variable (ici **NR**) au retour de la routine.

LOCBAS

```
100  D$ = CHR$ (4): PRINT D$"PR£3": PRINT : HOME
110  PRINT D$"BLOAD LOCNBR"
120  VTAB 20: HTAB 1: CALL - 958: PRINT "HTAB, VTAB, NOMBRE DE CHIFFRES"
125  VTAB 24: POKE 34,23: HOME
130  INPUT "—» H,V,N ";H,V,N
140  IF H < 1 OR H + N > 80 THEN 130
150  IF V < 1 OR V > 24 THEN 130
160  IF NOT N OR N > 11 THEN 130
170  TEXT : HOME
180  CALL 768,V,H,N,NR
190  PRINT : PRINT "VALEUR DANS (NR) : ";NR
200  VTAB 23: PRINT "(M)ENU DE DISQUETTE (E)NCORE (F)IN ";: GET R$
210  VTAB 20: PRINT : IF R$ = "M" OR R$ = "m" THEN PRINT D$"RUN MENU"
220  HOME : IF R$ = "E" OR R$ = "e" THEN 120
230  IF R$ = "F" OR R$ = "f" THEN END
240  GOTO 200
```

LOCNBR SUR 80 COLONNES

VTAB

```
300 :  20  F5  E6     JSR  $E6F5
303 :  CA             DEX
304 :  86  25         STX  $25
306 :  20  22  FC     JSR  $FC22
```

On passe par GETBYTC pour récupérer VTAB. Le résultat est dans X qui est décrémenté (0 à 23).

VTAB place le curseur sur la ligne indiquée dans CV ($25).

HTAB

```
309 :  20  F5  E6     JSR  $E6F5
30C :  CA             DEX
30D :  86  06         STX  $06
30F :  8E  7B  05     STX  $057B
```

Même processus pour HTAB qui est sauvegardé dans $6.

HTAB (toujours moins un : 0 à 79).

LONGUEUR DU MOT

```
312 :  20  F5  E6     JSR  $E6F5
315 :  8A             TXA
316 :  18             CLC
317 :  65  06         ADC  $06
319 :  85  07         STA  $07
```

Longueur du mot passée dans A.

On aura ainsi, en $7, la limite à ne pas dépasser.

(suite page 36)

MISE EN PLACE DU CURSEUR

```
31B : 20 9C FC   JSR $FC9C   CLREOL efface la fin de la ligne.
31E : A9 FF      LDA £$FF    Pavé tramé comme curseur.
320 : 20 ED FD   JSR $FDED
323 : CE 7B 05   DEC $057B   Attention ! il faut revenir sur la position !
```

SAISIE

```
326 : 2C 10 C0   BIT $C010   Clavier à zéro.
329 : AD 00 C0   LDA $C000   Lecture du clavier.
32C : 10 FB      BPL $0329   Si aucune touche n'a été pressée, on attend.
32E : 2C 10 C0   BIT $C010   Remise à zéro.
331 : C9 8D      CMP £$8D    Est-ce RETURN ?
333 : F0 1B      BEQ $0350   Si oui, saut : pour ce nombre, on a terminé.
335 : C9 88      CMP £$88    Est-ce la flèche arrière ?
337 : D0 0C      BNE $0345   Non : saut.
```

FLÈCHE ARRIÈRE

```
339 : AD 7B 05   LDA $057B   Si on est égal à HTAB, on ne doit plus reculer.
33C : C5 06      CMP $06
33E : F0 E6      BEQ $0326
340 : CE 7B 05   DEC $057B   Sinon, d'accord.
343 : 10 D6      BPL $031B   Retour jusqu'à l'effacement.
```

CONTRÔLE CARACTÈRES

```
345 : C9 B0      CMP £$B0    Plus petit que 0 refusé.
347 : 90 DD      BCC $0326
349 : C9 BA      CMP £$BA    Plus grand ou égal ":", refusé itou.
34B : B0 D9      BCS $0326
```

AFFICHAGE

```
34D : 20 ED FD   JSR $FDED   COUT fait son travail habituel.
350 : 48         PHA         Caractère sur la pile.
351 : AE 7B 05   LDX $057B   Position horizontale dans X.
354 : 8A         TXA
355 : 38         SEC         On a maintenant la valeur de Y
356 : E5 06      SBC $06     pour stocker le caractère, plus loin.
358 : A8         TAY
359 : 68         PLA         Récupération du caractère.
35A : C9 8D      CMP £$8D    Si ce n'est pas RETURN, on saute.
35C : D0 04      BNE $0362
```

0 EN FIN DE CHAÎNE

```
35E : A9 00      LDA £$00    0 dans l'accumulateur.
360 : C8         INY         Incrémentation normale de Y.
361 : E8         INX
362 : E4 07      CPX $07     Attention ! si l'on a atteint la longueur limite, il va falloir
364 : 90 01      BCC $0367   ajuster Y !
366 : C8         INY         C'est fait !
```

STOCKAGE DU CARACTÈRE

```
367 : 29 3F      AND £$3F    Où B1 devient 31, etc.
369 : 99 99 03   STA $0399,Y C'est ici qu'il nous faut la vraie valeur de Y.
36C : C9 00      CMP £$00    Si l'accumulateur contient 0, terminé.
36E : F0 06      BEQ $0376
```

```
370 : E4 07      CPX $07
372 : 90 AA      BCC $031E
374 : B0 AD      BCS $0323
```

Si on est plus petit que la limite, vers curseur.

Si on est plus grand ou égal, vers DEC.

TRANSFERT DANS LA VARIABLE

```
376 : 20 BE DE   JSR $DEBE
379 : 20 E3 DF   JSR $DFE3
37C : A5 B8      LDA $B8
37E : 48         PHA
37F : A5 B9      LDA $B9
381 : 48         PHA
382 : A9 9A      LDA £$9A
384 : 85 B8      STA $B8
386 : A9 03      LDA £$03
388 : 85 B9      STA $B9
38A : 20 67 DD   JSR $DD67
38D : 68         PLA
38E : 85 B9      STA $B9
390 : 68         PLA
391 : 85 B8      STA $B8
393 : A6 83      LDX $83
395 : A4 84      LDY $84
397 : 4C 2D EB   JMP $EB2D
```

CHKCOM teste la virgule et PTRGET cherche la variable.

Sauvegarde de l'adresse du dernier caractère obtenu par CHRGET... sinon on se plante.

On indique à TXTPTR l'adresse du nombre saisi (il se termine par un 0).

FRMNUM va évaluer cette formule (pointée par TXTPTR) et mettre le résultat dans FAC.

Rendons à TXTPTR ce qui n'est pas à Nestor !

Ça, c'est l'adresse de notre variable.
Elle dort ici depuis que PTRGET l'a trouvée.
MOVMF transfère FAC à l'adresse indiquée. TERMINÉ.

LOCNBR SUR 40 COLONNES

Pour fonctionner sur un écran de 40 colonnes, **LOCNBR** doit au préalable être corrigé par le petit patch ci-contre. En fait, on se contente de remplacer l'adresse $57B par $24 (CH = position horizontale du curseur). La routine ainsi obtenue est mémorisée sous le titre **LOCNBR40**.

```
250 REM Patch pour 40 colonnes (+40 au lieu de 80 à
    la ligne Basic 140)
260 FOR I = 1 TO 15: READ A,R: POKE A,R: NEXT :
    PRINT D$"BSAVE LOCNBR40,A$300,L154"
270 DATA 783,134,784,36,785,234,803,198,804,36,805,
    234,825,165,826,36,827,234,832,198,833,36,834,
    234,849,166,850,36,851,234
```

LET

Que fait cette routine de l'Applesoft ? elle affecte la valeur trouvée après le signe = à la variable qui est pointée par CHRGET. Voici un moyen de l'utiliser...

```
300 : 20 B1 00      JSR $00B1
303 : 4C 46 DA      JMP $DA46
```

```
100 TEXT : HOME : GOSUB 180
110 CALL 768,A = 7999
120 PRINT A
130 LIST 110,120
140 PRINT : PRINT "300: 20 B1 00   JSR 00B1    ; CHRGET"
150 PRINT "303: 4C 46 DA   JMP DA46   ; LET"
160 VTAB 23: PRINT "(M)ENU DE DISQUETTE ";: GET R$: VTAB 22: PRINT : IF R$ = "M" OR R$ = "m"
    THEN PRINT CHR$ (4)"RUN MENU"
170 END
180 POKE 768,32: POKE 769,177: POKE 770,0: POKE 771,76: POKE 772,70: POKE 773,218: RETURN
```

FOUT

Que fait FOUT ($ED34) ? Cette routine utilitaire crée tout simplement une chaîne de chiffres dans FBUFFR (à partir de $100). Cette chaîne correspond à la valeur de FAC. Pour contrôler le fonctionnement de FOUT, nous plaçons un nombre dans FAC par l'intermédiaire de CHKCOM ($DEBE) et FRMNUM ($DD67).
A la sortie de FOUT, Y et A pointent sur la chaîne. On a donc $1 dans Y et 0 dans A, d'où le DEY de notre petite routine... qui met Y à 0. On aurait pu le laisser à 1 et lire par LDA $DFF,Y.
Attention ! FOUT modifie FAC. Bien entendu, l'Applesoft a prévu une routine qui remplace notre affichage octet par octet : c'est STROUT ($DB3A) qui affiche une chaîne pointée par Y,A... exactement ce qu'il nous faut (notez que la chaîne lue dans FBUFFR se termine par un zéro).

```
100  TEXT : HOME : PRINT CHR$ (21): GOSUB
     200
110  INVERSE : PRINT "QUE FAIT FOUT. ROU
     TINE UTILITAIRE ED34 ?": NORMAL
120  PRINT "Plaçons la valeur 123 dans FAC:"
130  LIST 50: PRINT
140  CALL 768,123
150  PRINT : CALL 790
160  VTAB 22: PRINT
170  VTAB 24: PRINT "(M)ENU DE DIS
     QUETTE ";: GET R$: VTAB 22: PRINT
180  IF R$ = "M" OR R$ = "m" THEN PRINT
     CHR$ (4)"RUN MENU"
190  END
200  FOR I = 768 TO 820: READ R: POKE I,R:
     NEXT : RETURN
210  DATA  32,190,222,32,103,221,32,52,237,
     136,185,0,1,240,6,32,237,253,200,208,
     245,96
220  REM Routine d'affichage du programme
     LM ci-dessus
230  DATA  169,10,133,6,160,0,162,3,134,59,
     132,58,32,208,248,24,165,47,26,101,58,
     168,144,2,230,59,198,6,208,236,96
```

```
300 :  20  BE  DE    JSR  DEBE      ┐ Transmission
303 :  20  67  DD    JSR  DD67      ┘ du nombre à FAC.
306 :  20  34  ED    JSR  ED34        FOUT (Y=1 et A=0)
309 :  88            DEY              Y=0.
30A :  B9  00  01    LDA  0100,Y    ┐
30D :  F0  06        BEQ  0315 é+06è  Affichage
30F :  20  ED  FD    JSR  FDED        en mode inverse.
312 :  C8            INY
313 :  D0  F5        BNE  030A é-0Bè┘
315 :  60            RTS              Retour.
```

Même résultat, mais en mode normal

```
300 : 20 BE DE   JSR DEBE
303 : 20 67 DD   JSR DD67
306 : 20 34 ED   JSR ED34
309 : 4C 3A DB   JSR DB3A
```

CATIMP1

CATALOG (80 colonnes) ou CAT (40 colonnes), pour tout slot compris entre 4 et 7 (drive 1 ou 2), avec ou sans impression.

SYNTAXE : BRUN CATIMP1 (en mode direct ou à partir d'un programme Basic) — BLOAD CATIMP1, puis CALL 768 possible à partir d'un programme, mais pas en mode direct (fonctionne, mais avec SYNTAX ERROR).

CURSEUR

```
300 : A9 16      LDA £16
302 : 85 25      STA 25
304 : 20 42 FC   JSR FC42
```

CV = $16 (VTAB 23).

CLREOP (le CALL - 958 du Basic) efface les lignes 23 et 24.

AFFICHAGE TEXTE 1

```
307 : A0 03      LDY £03
309 : A9 86      LDA £86
30B : 20 3A DB   JSR DB3A
```

Adresse du texte n°1 dans Y et A.
STROUT affiche la chaîne qui se termine par 0.

SLOT-DRIVE :

SLOT (4 à 7)

```
30E : 20 35 FD   JSR FD35
311 : C9 B4      CMP £B4
313 : 90 F9      BCC 030E
315 : C9 B8      CMP £B8
317 : B0 F5      BCS 030E
319 : 8D A6 03   STA 03A6
31C : A2 02      LDX £02
31E : 20 4C F9   JSR F94C
```

RDCHAR attend un caractère.

Plus petit que 4, il est refusé.

Plus grand ou égal à 8, il est refusé aussi.

Numéro du SLOT stocké à sa place dans TEXTE3.

PRBL3 affiche A plus un espace (x - 1).

DRIVE 1 ou 2

```
321 : 20 35 FD   JSR FD35
324 : C9 B1      CMP £B1
326 : 90 F9      BCC 0321
328 : C9 B3      CMP £B3
32A : B0 F5      BCS 0321
32C : 8D A9 03   STA 03A9
32F : A2 03      LDX £03
331 : 20 4C F9   JSR F94C
```

RDCHAR encore.

Plus petit que 1, refusé.

Plus grand que 3, refusé.

Numéro du DRIVE stocké à sa place dans TEXTE 3.

PRBL3 affiche A plus deux espaces (x - 1).

AFFICHAGE TEXTE 2

```
334 : A0 03      LDY £03
336 : A9 93      LDA £93
338 : 20 3A DB   JSR DB3A
```

Adresse du texte.

STROUT fait son travail.

(O) = IMP:

(O)UI POUR IMPRIMER ?

```
33B : 20 35 FD   JSR FD35
33E : C9 CF      CMP £CF
340 : F0 04      BEQ 0346
342 : C9 EF      CMP £EF
344 : D0 04      BNE 034A
```

RDCHAR.

Si c'est le O majuscule, saut.

Si ce n'est pas le o minuscule, saut.

(suite page 40)

```
346 : A9 01      LDA £01
348 : D0 02      BNE 034C
34A : A9 00      LDA £00
34C : 48         PHA
34D : A5 36      LDA 36
34F : 85 06      STA 06
351 : A5 37      LDA 37
353 : 85 07      STA 07
```

Accumulateur = 1 pour "O" ou "o" et saut.

Sinon A = 0.
En attente sur la pile.

Pour revenir au Basic après impression éventuelle.

IMP OU NON ?

```
355 : 68         PLA
356 : F0 03      BEQ 035B
358 : 20 95 FE   JSR FE95
```

On dépile vers A.
Si c'est 0, on n'imprime pas : saut.
A = 1. On a PR£1 (il faut l'imprimante en 1).

INSTRUCTION DANS LE BUFFER

```
35B : A0 00      LDY £00
35D : B9 9D 03   LDA 039D,Y
360 : 99 00 02   STA 0200,Y
363 : C8         INY
364 : C9 8D      CMP £8D
366 : D0 F5      BNE 035D
```

Le texte n°3
est transféré
dans le buffer.

CATALOG,S6,D1

40 OU 80 COLONNES ?

```
368 : AD 1F C0   LDA C01F
36B : 30 0A      BMI 0377
36D : A0 03      LDY £03
36F : A9 A0      LDA £A0
371 : 99 03 02   STA 0203,Y
374 : 88         DEY
375 : 10 FA      BPL 0371
377 : 20 03 BE   JSR BE03
37A : 20 66 FC   JSR FC66
37D : A5 06      LDA 06
37F : 85 36      STA 36
381 : A5 07      LDA 07
383 : 85 37      STA 37
385 : 60         RTS
```

Si on est en 80 colonnes, saut.

On remplace **ALOG**, dans le buffer, par autant d'espaces $A0.

Commande externe de ProDOS.
LF.

CSW.

et CSWH rétablis.

FIN DE PROGRAMME.

TEXTES

```
386 : 53 4C 4F 54 2D 44 52 49 56 45 3A 20 00
```

SLOT-DRIVE :

```
393 : 28 4D 29 3D 49 4D 50 3A 20 00
```

(M) = IMP :

```
39D : C3 C1 D4 C1 CC CF C7 AC D3 B5 AC C4 B1 A0 8D
```

CATALOG,S6,D1

et RETURN

RACINE CARRÉE

CALL 768,100...

```
300 : 20 BE DE   JSR $DEBE   TESTE LA VIRGULE.
303 : 20 67 DD   JSR $DD67   MET LE NOMBRE DANS FAC.
306 : 20 8D EE   JSR $EE8D   CALCULE LA RACINE CARRÉE.
309 : 4C 2E ED   JMP $ED2E   AFFICHE LA VALEUR TROUVÉE.
```

Clément RENARD

GSDATE0

UTILISATION BLOAD GSDATE0, puis CALL 768 ou BRUN GSDATE0.

```
300 :  18              CLC
301 :  FB              XCE          ] Passage en mode natif pur.
302 :  C2 30           REP  £30     ]
304 :  A0 00 00        LDY  £0000     
307 :  A9 25 03        LDA  0325      Buffer à partir de $325.
30A :  5A              PHY
30B :  48              PHA
30C :  A2 03 0F        LDX  £0F03     ReadAsciiTime : $0F03.
30F :  22 00 00 E1     JSL  E10000
313 :  38              SEC          ] Passage en mode émulation.
314 :  FB              XCE          ]
315 :  A9 17           LDA  £17     ] Adresse de la ligne 24.
317 :  20 47 F8        JSR  F847    ]
31A :  A0 13           LDY  £13       19 décimal dans Y (0 à 19).
31C :  B9 25 03        LDA  0325,Y    Récupération.
31F :  91 26           STA  (26),Y    Affichage par POKE.
321 :  88              DEY            Y = Y - 1.
322 :  10 F8           BPL  031C      Oui jusqu'à Y = 0 inclus.
324 :  60              RTS            FIN.
```

GSDATE2

UTILISATION CALL 768, A$: PRINT A$

(suite page 42)

PRÉPARATION (variable - buffer)

```
300 :  A9 14           LDA  £14     ] Longueur de la variable-date : $14 (20). STRINI réserve de
302 :  20 D5 E3        JSR  E3D5    ] l'espace pour une chaîne et crée un descripteur dans DSCTMP.
```

```
305 : A5 9E      LDA 9E
307 : 8D 17 03   STA 0317
30A : A5 9F      LDA 9F
30C : 8D 18 03   STA 0318
```

Adresse de l'emplacement de la chaîne installée dans le programme.

ReadAsciiTime

```
30F : 18         CLC
310 : FB         XCE
```

Passage en mode natif pur.

```
311 : C2 30      REP £30
313 : A0 00 00   LDY £0000
316 : A9 00 00   LDA $0000
319 : 5A         PHY
31A : 48         PHA
31B : A2 03 0F   LDX £0F03
31E : 22 00 00 E1 JSL E10000
```

Adresse installée au début de la routine.

ReadAsciiTime : $0F03.

```
322 : 38         SEC
323 : FB         XCE
```

Mode émulation.

RÉCUPÉRATION

```
324 : 20 BE DE   JSR DEBE
327 : 20 E3 DF   JSR DFE3
```

Test de la virgule et recherche de la variable (adresse dans A et Y).

```
32A : 85 85      STA 85
32C : 84 86      STY 86
```

Adresse placée dans FORPNT (pointeur pour COPY).

```
32E : A9 9D      LDA £9D
330 : A0 00      LDY £00
```

Adresse du descripteur actuel.

```
332 : 4C B7 DA   JMP DAB7
```

COPY déplace la chaîne temporaire et libère sa place.

GSCAT3

Curiosité : vous envoyez "=T" et un CTRL-Q ou Y... ou C qui vous fait sortir du Moniteur. Peut être intéressant lorsque vous programmez en Basic. Vous pouvez ainsi obtenir l'heure, en tapant **CALL 768** (la routine est relogeable). Si vous désirez consulter l'heure très souvent, fournissez l'adresse de la routine à l'Ampersand : un simple & vous donnera l'heure.

```
300 : A0 04      LDY £$04       Quatre tours de boucle.
302 : B9 0F 03   LDA $030F,Y    Lecture d'un caractère.
305 : 99 FF 01   STA $01FF,Y    Ecriture dans le buffer.
308 : 88         DEY            Y = Y - 1.
309 : D0 F7      BNE $0302      Oui jusqu'à 1 inclus.
30B : 84 B8      STY $B8        0 dans TXTPTR.
30D : 4C 70 FF   JMP $FF70      SCAN lit le buffer et exécute les instructions.
310 : BD D4 91
313 : 8D                        = T, CTRL-Q et RETURN
```

■

APPLE IIGS

Stephan HADINGER

ACCESSOIRE : ACCENT

L'un des défauts de jeunesse de l'Apple *IIGS* est que, pour presque toutes les applications de l'environnement Desktop, il est impossible de taper des minuscules accentuées et autres caractères internationaux.

Pourtant ces caractères sont bel et bien présents dans les jeux de caractères, mais leurs codes ASCII sont supérieurs à 128 ($80 en hexa), ce qui les rend inaccessibles au clavier. L'accessoire du bureau ACCENT résoud ce problème et permet ainsi de rentrer tous les caractères ayant des codes de 128 à 255 ($80 à $FF).

LE PROGRAMME

Vous pouvez :

— soit assembler ACCENT avec l'assembleur ORCA/M de la disquette Workshop en faisant :

- ASSEMBLE ACCENT.SRC
- LINK ACCENT KEEP = ACCENT
- FILETYPE ACCENT NDA

— soit, pour ceux qui ne disposent pas de Workshop, entrer dans le moniteur le récapitulatif hexa, puis le sauver par :

- CREATE ACCENT,T$B8
- BSAVE ACCENT,A$2000,L2048,T$B8

Ensuite, placez l'accessoire dans le sous-directory SYSTEM/DESK.ACCS/ de l'application sur laquelle vous voulez l'installer. Lancez cette application. L'accessoire "Minuscules Accentuées" sera désormais accessible par le menu Pomme.

UTILISATION

Dès que vous sélectionnez l'accessoire, il devient actif et une fenêtre de présentation apparaît ; il le restera même si la fenêtre n'est pas au premier plan, et ceci jusqu'à ce que vous le fermiez en cliquant dans la case de fermeture.

Si la présence de la fenêtre vous gêne, cliquez dans le bouton "Cacher fenêtre" et elle disparaît, laissant néanmoins l'accessoire actif. Pour la faire réapparaître, appuyez simultanément sur Pomme et Option, puis sur Esc SANS RELÂCHER les deux autres touches.

Lorsque vous voulez entrer un caractère spécial, appuyez simultanément sur Pomme et Option et RELÂCHEZ-LES. Dès lors, l'application est suspendue (le curseur ne clignote pas et la souris ne bouge plus car les interruptions sont inhibées) ; reportez-vous à la table de conversion (page 53) et tapez sur la touche correspondant au caractère souhaité ; celui-ci apparaîtra alors.

Si vous avez appuyé sur Pomme et Option et que vous désirez annuler votre demande de caractère spécial, appuyez sur une touche quelconque (sauf Esc) SANS RELÂCHER Pomme et Option. L'accessoire rend alors le contrôle à l'application.

Cet accessoire de bureau est compatible avec tous les programmes, fonctionnant sous l'environnement Desktop, actuellement disponibles ; cependant, avec GS/WRITE 1.0. la fenêtre de présentation reste vide, bien que l'accessoire fonctionne correctement.

N.B. : Certains jeux de caractères comme *Shaston* ou *Venice* ne contiennent pas tous les caractères spéciaux. À vous de les découvrir... Attention ! dans certains programmes comme *GraphicWriter* ou *VS/Draw*, les jeux de caractères ont été "francisés", mais seules les minuscules accentuées du clavier AZERTY y sont présentes. Leur utilisation est rendue inutile par le présent accessoire et il vaut même mieux ne pas les utiliser car les caractères crochets et accolades en sont absents.

```
*
*       minuscules accentuées en accessoire de bureau (NDA)
*

        absaddr on
        65816 on
        keep  ACCENT
        mcopy ACCENT.MACROS
        case  off

IDSection START
        dc    i4'OpenACC'               ; prologue de l'accessoire
        dc    i4'CloseACC'
        dc    i4'ActionACC'
        dc    i4'InitACC'
        dc    i2'0'                     ; rafraichissement maximal
        dc    i2'$0143'                 ; Activate, Update & MouseDown events
        dc    c'  Minuscules Accentu'   ; titre de l'accessoire
        dc    i1'$8E'                   ; e accent aigu
        dc    c'esçH**'
        dc    i1'13'
        END

OpenACC START                           ; ouverture de l'accessoire
        using ACCData

        lda   >ACCActive                ; teste si déjà ouvert
        beq   Cont
        rtl

Cont    PushLong £0
        PushLong £ACCParams
        _NewWindow                      ; ouvre la nouvelle fentre
        plx
        pla
        sta   6,s
        sta   >ACCWinPtr+2
        txa
        sta   4,s
        sta   >ACCWinPtr
        PushLong £0
        PushLong >ACCWinPtr
        PushLong £BRect
        PushLong £CtrlTitle
        PushWord £0
        PushWord £0
        PushLong £0
        PushLong £0
        PushLong £0
        PushLong £0
        _NewControl                     ; crée le bouton de contrôle
        PullLong >CtrlPtr
        PushLong >ACCWinPtr
        _SetSysWindow                   ; déclare la fenêtre comme une
;                                         fenêtre système
```

Ceux qui le souhaitent peuvent parfaitement remplacer, sur leur copie de travail, les fontes bidouillées (pardon, francisées) en les remplaçant par leurs homonymes américaines qui traînent un peu partout. Simplement ils devront utiliser ACCENT pour saisir les é,è et autres ù. A ce prix, ils auront la possibilité de saisir crochets, accolades et autres gâteries. **Y. K.**

ATTENTION ! *dans ces pages, nous avons parfois présenté les instructions sur 2 colonnes. Tenez alors compte des petites flèches.*

```
         lda   £$8000
         sta   >ACCActive
         lda   £0
         sta   >runstate
         rtl

ACCParams anop                          ; Paramètres de la fenêtre
         dc    i2'78'
         dc    i2'%1100000010100000'
         dc    i4'ACCTitle'
         dc    i4'0'
         dc    i2'0,0,0,0'
         dc    i4'0'
         dc    i2'0,0,0,0,0,0,0,0,0,0'
         dc    i4'0'
         dc    i2'0'
         dc    i4'0,0,0'
         dc    i'50,70,122,305'          ; rectangle d'affichage
         dc    i4'$FFFFFFFF'             ; fenêtre au premier plan
         dc    i4'0'

CtrlTitle dc   i1'14'                   ; texte du bouton de contrôle
         dc    c'Cacher fen'
         dc    i1'$90'                   ; e accent circonflexe
         dc    c'tre'
BRect    dc    i2'48,47,61,188'          ; rectangle d'affichage
         END

ACCData  DATA                           ; variables diverses
runstate dc    i'0'
ACCActive dc   i'0'                     ; accessoire actif ?
ACCTitle str   'Accents'                ; titre de la fenêtre
ACCWinPtr ds   4
CtrlPtr  ds    4
where    ds    4                        ; lieu de l'événement
CtrlHndl ds    4
Rectangle dc   i'0,0,72,235'            ; rectangle de la fenêtre
;                                         (coordonnées locales)

ligne1   dc    c'Minuscules accentu'     ; texte affiché
         dc    i1'$8E'
         dc    c'es,(c) 1987'
         dc    i1'0'
ligne2   dc    c'S. HADINGER & TREMPLIN MICRO'
         dc    i1'0'
ligne3   dc    c'appuyez sur '
         dc    i1'$11'                   ; caractère Pomme
         dc    c' et Option,'
         dc    i1'0'
ligne4   dc    c'relachez, et tapez une lettre.'
         dc    i1'0'
ligne5   dc    i1'$11'
         dc    c'-Option-Esc fait r'
         dc    i1'$8E'
         dc    c'appara'
         dc    i1'$94'
```

A ceux qui saisissent en HEXA, je conseille de faire la sauvegarde en mode BIN sous un nom provisoire. AKSENT fera l'affaire. Cela leur permettra d'utiliser SIGNATURE pour contrôler le travail de saisie. Lorsque les contrôles seront satisfaisants, il leur suffira de taper :

BLOAD AKSENT CREATE ACCENT,T$B8
BSAVE ACCENT,A$2000,L2048,T$B8 **V. K.**

(SUITE PAGE 46)

```
        dc    c'tre.'
        dc    i1'0'
        END

CloseACC START
        using ACCData

        lda   >ACCActive
        beq   Ignore
        PushLong >ACCWinPtr
        _CloseWindow                    ; fermeture de la fenêtre
        lda   £0
        sta   >ACCActive
Ignore  rtl
        END

ActionACC START                         ; gestion des événements
        using ACCData

        phb
        phk
        plb
        phy
        phx
        asl   a
        tax
        jsr   (ActionTable,x)
        pla
        pla
        plb
        rtl

ActionTable anop
        dc    i'Ignore'
        dc    i'ActionEvent'            ; événement standard
        dc    i'ActionRun'              ; rafraichissement de l'accessoire
        dc    i'ActionCursor'
        dc    i'ActionMenu'
        dc    i'ActionUndo'
        dc    i'ActionCut'
        dc    i'ActionCopy'
        dc    i'ActionPaste'
        dc    i'ActionCLear'

ActionCursor    anop                    ; ignore tous les autres événements
ActionMenu      anop
ActionUndo      anop
ActionCut       anop
ActionCopy      anop
ActionPaste     anop
ActionClear     anop

Ignore  anop
        rts
        END
```

Quand on dispose de l'assembleur, on peut éviter une commande en faisant :

ASML ACCENT.SRC FILETYPE ACCENT NDA

Il est inutile de saisir le fichier macros, l'assembleur fera le travail tout seul si vous lui dites poliment :

MACGEN ACCENT.SRC ACCENT.MACROS 2/AINCLUDE/M16.=

N'omettez aucun caractère, espaces compris. **Y. K.**

Ceux qui disposent d'un système d'exploitation 2.Ø, 3.Ø ou 3.1 peuvent employer des fichiers compactés. Pour compacter un fichier, il faut avoir APW 1.Ø et lui envoyer la commande :

COMPACT ACCENT -Ø ACCENTC

Je vous laisse le soin de manipuler les fichiers par la suite. ACCENT comporte 2Ø48 octets, ACCENT.C n'en contient que 1392.

ATTENTION, le système 1.Ø ne sait pas manipuler les fichiers compactés, mais vous devriez penser à quitter cette vieillerie. **Y. K.**

HADINGER mentionne GS.WRITE 1.Ø. Je lui pardonne car il a écrit cet article il y a quelques mois. Par contre je ne veux plus entendre parler de cette version. Au lieu de râler contre ses bugs, demandez à votre concessionnaire la mise à jour 2.Ø (qui n'est d'ailleurs pas parfaite). **Y. K.**

```
ActionEvent START
          using ACCData
evtptr    equ   5

          phd
          tsc
          tcd
          ldy   #$0A
          lda   [evtptr],y
          sta   where
          iny   ▶

Activate  anop
          PushLong #0
          _GetPort                    ; sauve le port
          PushLong ACCWinPtr
          _SetPort                    ; utilise le port de la fenêtre
          PushLong ACCWinPtr
          _DrawControls               ; dessine le bouton
          _SetPort
Ignore2   rts

MouseDown PushWord #0
          PushLong #CtrlHndl
          PushLong where
          PushLong ACCWinPtr
          _FindControl                ; a-t-on cliqué sur le bouton
          lda   1,s
          beq   mousefin

          PushWord #0
          PushLong where
          PushLong #0
          PushLong CtrlHndl
          _TrackControl               ; gestion du bouton
          pla
          cmp   1,s
          bne   mousefin

          PushLong ACCWinPtr
          _HideWindow                 ; cache la fenêtre
;                                       mais l'accessoire reste actif
mousefin  pla
          rts

Update    pushlong ACCWinPtr
          _BeginUpdate
          PushLong #Rectangle
          _EraseRect                  ; efface la fenêtre
          PushWord #7
          PushWord #10
          _MoveTo
          PushLong #ligne1
          _DrawCString                ; affiche le texte
```

```
          iny
          lda   [evtptr],y
          sta   where+2
          lda   [evtptr]
          pld
          cmp   #6
          bne   Cont
          brl   Update
Cont      cmp   #1
          beq   MouseDown
          cmp   #8
          bne   Ignore2
```

(SUITE PAGE 48)

```
        PushWord £7
        PushWord £20
        _MoveTo
        PushLong £ligne2
        _DrawCString

        PushWord £7
        PushWord £33
        _MoveTo
        PushLong £ligne3
        _DrawCString

        PushWord £7
        PushWord £43 ▶
                                        _MoveTo
                                        PushLong £ligne4
                                        _DrawCString

                                        PushWord £7
                                        PushWord £70
                                        _MoveTo
                                        PushLong £ligne5
                                        _DrawCString

                                        PushWord £0
                                        PushWord £22
                                        _MoveTo
                                        PushWord £235
                                        PushWord £22
        _LineTo                     ; dessine les lignes de separation

        PushWord £0
        PushWord £45
        _MoveTo
        PushWord £235
        PushWord £45
        _LineTo

        PushLong ACCWinPtr
        _DrawControls               ; dessine le bouton

        pushlong ACCWinPtr
        _EndUpdate
        rts
        END


InitACC START                       ; initialisation de l'accessoire
        using ACCData
        lda   £0
        sta   >ACCActive
        sta   >runstate
        rtl
        END


ActionRun START                     ; rafraichissement
        using ACCData

        phb
        phk
        plb
        lda   runstate              ; précédente demande annulée ?
        bne   libre
        shortm
        lda   >$E0C061              ; est-ce que Pomme ET Option
        bpl   end                   ; sont appuyées
        lda   >$E0C062
        bpl   end
```

```
            php
            sei
loop1       lda   >$E0C061          ; attente d'une touche
            bpl   ok
            lda   >$E0C062
            bpl   ok
            lda   >$E0C000
            bpl   loop1
            cmp   £$9B              ; si ESC tapé...
            bne   Cont2
            long
            PushLong ACCWinPtr
            _ShowWindow             ; ...réaffiche la fenêtre...
            PushLong ACCWinPtr
            _SelectWindow           ; ...et la place au premier plan
Cont2       inc   runstate          ; annulation demandée
nogood      shortm
            lda   >$E0C010
            plp

end         long
            plb
            rts

ok          anop
            longa off
loop2       lda   >$E0C000
            _PostEvent
            pla
            bra   nogood

libre       shortm
            lda   >$E0C061
            bmi   libend
            lda   >$E0C062
            bmi   libend
            stz   runstate
libend      long
            plb
            rts
            END
```

```
            bpl   loop2
            sec
            sbc   £$40
            ora   £$80
            long
            and   £$00FF
            PushWord £0
            PushWord £3
            PushWord £0
            pha
; envoie l'événement clavier
; à l'application

; attend que Pomme ET Option
; soient relâchées
```

ATTENTION ! *dans ces pages, nous avons parfois présenté les instructions sur 2 colonnes. Tenez alors compte des petites flèches.*

ACCENT.MACROS

```
 1  MACRO
 2 &lab pulllong &addr1,
 3 &lab ANOP          &addr2
 4  AIF C:&addr1=0,.a
 5  AIF C:&addr2=0,.b
 6  LCLC &C
 7 &C AMID &addr1,1,1
 8  AIF "&C"="°",.zeropage
 9  pullword &addr1
10  sta &addr2
11  pullword &addr1+2
12  sta &addr2+2
13  MEXIT
14 .a
15  pullword
16  pullword
17  MEXIT
18 .b
19  pullword &addr1
20  pullword &addr1+2
21  MEXIT
22 .zeropage
23  ldy £&addr2
24  pullword &addr1,y
25  ldy £&addr2+2
26  pullword &addr1,y
27  MEND
28  MACRO
29 &lab pullword &sysopr
30 &lab ANOP
31  pla
32  AIF c:&sysopr=0,.end
33  sta &sysopr
```

```
34 .end
35  MEND
36  MACRO
37 &lab pushlong &addr,&offset
38 &lab ANOP
39  LCLC &C
40  LCLC &REST
41 &C AMID &addr,1,1
42  AIF "&C"="£",.immediate
43  AIF "&C"="°",.zeropage
44  AIF C:&offset=0,.nooffset
45  AIF "&offset"="s",.stack
46  pushword &addr+2,&offset
47  pushword &addr,&offset
48  MEXIT
49 .nooffset
50  pushword &addr+2
51  pushword &addr
52  MEXIT
53 .immediate
54 &REST AMID &addr,2,L:&addr-1
55  dc I1'$F4',I2'(&REST)ù-16'
56  dc I1'$F4',I2'&REST'
57  MEXIT
58 .stack
59  pushword &addr+2,s
60  pushword &addr+2,s
61  MEXIT
62 .zeropage
63  ldy £&offset+2
64  pushword &addr,y
65  ldy £&offset
66  pushword &addr,y
67  MEND
68  MACRO
69 &lab pushword &sysopr
70 &lab ANOP
71  AIF c:&sysopr=0,.b
72  LCLC &C
73  LCLC &REST
74  LCLA &BL
75 &C AMID "&sysopr",1,1
76 &BL ASEARCH "&sysopr"," ",1
77  AIF &BL>0,.a
78 &BL SETA L:&sysopr+1
79 .a
80 &REST AMID "&sysopr",2,&BL-2
81  AIF ("&C"="£").AND.(S:LONGA),.immediate
82  lda &sysopr
83  pha
84  MEXIT
85 .b
86  pha
87  MEXIT
88 .immediate
89  dc I1'$F4',I2'&REST'
90  MEND
91  MACRO
92 &lab str &string
93 &lab dc i1'l:&string',
94  MEND          c'&string'
95  MACRO
96 &lab long
97 &lab ANOP
98  rep £%00110000
99  longa on
100  longi on
101  MEND
102  MACRO
103 &lab shortm
104 &lab ANOP
105  sep £%00100000
106  longa off
107  MEND
108  MACRO
109 &lab _NewControl
110 &lab ldx £$0910
111  jsl $E10000
112  MEND
```

FICHIER DE TYPE $B8

ACCENT

```
2000: 04 00 00 00 00 00 00 00 69 04 00 00 00 0A 04 01   F280
2010: 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00   7E01
2020: 00 00 01 00 00 00 00 00 2C 00 40 00 20 20 20 20   B6ED
2030: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20   1400
2040: F2 69 04 00 00 30 00 00 00 E1 01 00 00 00 02 00   5D73
2050: 00 C8 03 00 00 00 00 43 01 20 20 4D 69 6E 75 73   755B
2060: 63 75 6C 65 73 20 41 63 63 65 6E 74 75 8E 65 73   B465
2070: 5C 48 2A 2A 0D AF 2B 01 00 F0 01 6B F4 00 00 F4   E024
2080: 00 00 F4 00 00 F4 C4 00 A2 0E 09 22 00 00 E1 FA   D762
2090: 68 83 06 8F 37 01 00 BA 83 04 8F 35 01 00 F4 00   D882
20A0: 00 F4 00 00 AF 37 01 00 48 AF 35 01 00 48 F4 00   9F44
20B0: 00 F4 21 01 F4 00 00 F4 12 01 F4 00 00 F4 00 00   D2F9
20C0: F4 00 00 F4 00 00 F4 00 00 F4 00 00 F4 00 00 F4   FDB8
20D0: 00 00 F4 00 00 F4 00 00 A2 10 09 22 00 00 E1 68   BA0E
20E0: 8F 39 01 00 68 8F 3B 01 00 AF 37 01 00 48 AF 35   DB0F
20F0: 01 00 48 A2 0E 4B 22 00 00 E1 A9 00 80 8F 2B 01   C12B
2100: 00 A9 00 00 8F 29 01 00 6B 4E 00 A0 C0 2D 01 00   EAA9
2110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   0000
2120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   0000
2130: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   0000
2140: 00 00 00 00 00 00 00 32 00 46 00 7A 00 31 01 FF   1623
2150: FF FF FF 00 00 00 00 0E 43 61 63 68 65 72 20 66   06D7
2160: 65 6E 90 74 72 65 30 00 2F 00 3D 00 BC 00 00 00   D506
2170: 00 00 07 41 63 63 65 6E 74 73 00 00 00 00 00 00   15C8
2180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 48 00   B148
2190: EB 00 4D 69 6E 75 73 63 75 6C 65 73 20 61 63 63   E95A
21A0: 65 6E 74 75 8E 65 73 2C 28 63 29 20 31 39 38 37   31FB
```

```
113  MACRO
114 &lab _DrawControls
115 &lab ldx £$1010
116  jsl $E10000
117  MEND
118  MACRO
119 &lab _FindControl
120 &lab ldx £$1310
121  jsl $E10000
122  MEND
123  MACRO
124 &lab _TrackControl
125 &lab ldx £$1510
126  jsl $E10000
127  MEND
128  MACRO
129 &lab _PostEvent
130 &lab ldx £$1406
131  jsl $E10000
132  MEND
133  MACRO
134 &lab _NewWindow
135 &lab ldx £$090E
136  jsl $E10000
137  MEND
138  MACRO
139 &lab _CloseWindow
140 &lab ldx £$0B0E
141  jsl $E10000
142  MEND
143  MACRO
144 &lab _SelectWindow
145 &lab ldx £$110E
146  jsl $E10000
147  MEND
148  MACRO
149 &lab _HideWindow
150 &lab ldx £$120E
151  jsl $E10000
152  MEND
153  MACRO
154 &lab _ShowWindow
155 &lab ldx £$130E
156  jsl $E10000
157  MEND
158  MACRO
159 &lab _BeginUpdate
160 &lab ldx £$1E0E
161  jsl $E10000
162  MEND
163  MACRO
164 &lab _EndUpdate
165 &lab ldx £$1F0E
166  jsl $E10000
167  MEND  ▶
```

```
21B0: 00 53 2E 20 48 41 44 49 4E 47 45 52 20 26 20 54  A89D
21C0: 52 45 4D 50 4C 49 4E 20 4D 49 43 52 4F 00 61 70  2B82
21D0: 70 75 79 65 7A 20 73 75 72 20 11 20 65 74 20 4F  5750
21E0: 70 74 69 6F 6E 2C 00 72 65 6C 61 63 68 65 7A 2C  F9D0
21F0: 20 65 74 20 74 61 70 65 7A 20 75 6E 65 20 6C 65  9E96
2200: 74 74 72 65 2E 00 11 2D 4F 70 74 69 6F 6E 2D 45  BF16
2210: 73 63 20 66 61 69 74 20 72 8E 61 70 70 61 72 61  CB2F
2220: 94 74 72 65 2E 00 AF 2B 01 00 F0 18 AF 37 01 00  36D7
2230: 48 AF 35 01 00 48 A2 0E 0B 22 00 00 E1 A9 00 00  30DC
2240: 8F 2B 01 00 6B 8B 4B AB 5A DA 0A AA FC 0E 02 68  E603
2250: 68 AB 6B 22 02 23 02 D4 03 22 02 22 02 22 02 22  A22C
2260: 02 22 02 22 02 22 02 60 0B 3B 5B A0 0A 00 B7 05  42D5
2270: 8D 3D 01 C8 C8 B7 05 8D 3F 01 A7 05 2B C9 06 00  668A
2280: D0 03 82 97 00 C9 01 00 F0 38 C9 08 00 D0 32 F4  D9A5
2290: 00 00 F4 00 00 A2 04 1C 22 00 00 E1 AD 37 01 48  B4E6
22A0: AD 35 01 48 A2 04 1B 22 00 00 E1 AD 37 01 48 AD  69C9
22B0: 35 01 48 A2 10 10 22 00 00 E1 A2 04 1B 22 00 00  0626
22C0: E1 60 F4 00 00 F4 00 00 F4 41 01 AD 3F 01 48 AD  EE41
22D0: 3D 01 48 AD 37 01 48 AD 35 01 48 A2 10 13 22 00  90C5
22E0: 00 E1 A3 01 F0 34 F4 00 00 AD 3F 01 48 AD 3D 01  1FBD
22F0: 48 F4 00 00 F4 00 00 AD 43 01 48 AD 41 01 48 A2  BA42
2300: 10 15 22 00 00 E1 68 C3 01 D0 0F AD 37 01 48 AD  A60D
2310: 35 01 48 A2 0E 12 22 00 00 E1 68 60 AD 37 01 48  F438
2320: AD 35 01 48 A2 0E 1E 22 00 00 E1 F4 00 00 F4 45  6229
2330: 01 A2 04 55 22 00 00 E1 F4 07 00 F4 0A 00 A2 04  D69E
2340: 3A 22 00 00 E1 F4 00 00 F4 4D 01 A2 04 A6 22 00  1EE1
2350: 00 E1 F4 07 00 F4 14 00 A2 04 3A 22 00 00 E1 F4  C6BB
2360: 00 00 F4 6C 01 A2 04 A6 22 00 00 E1 F4 07 00 F4  C39F
2370: 21 00 A2 04 3A 22 00 00 E1 F4 00 00 F4 89 01 A2  0B18
2380: 04 A6 22 00 00 E1 F4 07 00 F4 2B 00 A2 04 3A 22  44C9
2390: 00 00 E1 F4 00 00 F4 A2 01 A2 04 A6 22 00 00 E1  E1BB
23A0: F4 07 00 F4 46 00 A2 04 3A 22 00 00 E1 F4 00 00  3C0C
23B0: F4 C1 01 A2 04 A6 22 00 00 E1 F4 00 00 F4 16 00  4803
23C0: A2 04 3A 22 00 00 E1 F4 EB 00 F4 16 00 A2 04 3C  EAAE
23D0: 22 00 00 E1 F4 00 00 F4 2D 00 A2 04 3A 22 00 00  EA1A
23E0: E1 F4 EB 00 F4 2D 00 A2 04 3C 22 00 00 E1 AD 37  47AA
23F0: 01 48 AD 35 01 48 A2 10 10 22 00 00 E1 AD 37 01  141E
2400: 48 AD 35 01 48 A2 0E 1F 22 00 00 E1 60 A9 00 00  904E
2410: 8F 2B 01 00 8F 29 01 00 6B 8B 4B AB AD 29 01 D0  F407
2420: 78 E2 20 AF 61 C0 E0 10 48 AF 62 C0 E0 10 42 08  6E8D
2430: 78 AF 61 C0 E0 10 3E AF 62 C0 E0 10 38 AF 00 C0  24DE
2440: E0 10 EE C9 9B D0 20 C2 30 AD 37 01 48 AD 35 01  CA34
2450: 48 A2 0E 13 22 00 00 E1 AD 37 01 48 AD 35 01 48  7666
2460: A2 0E 11 22 00 00 E1 EE 29 01 E2 20 AF 10 C0 E0  FE3D
2470: 28 C2 30 AB 60 AF 00 C0 E0 10 FA 38 E9 40 09 80  2868
2480: C2 30 29 FF 00 F4 00 00 F4 03 00 F4 00 00 48 A2  8FE3
2490: 06 14 22 00 00 E1 68 80 D1 E2 20 AF 61 C0 E0 30  8AB8
24A0: 09 AF 62 C0 E0 30 03 9C 29 01 C2 30 AB 60 F5 04  0DA9
24B0: 00 00 00 30 00 F5 04 00 04 00 E1 01 F5 04 00 08  E910
24C0: 00 00 02 F5 04 00 0C 00 C8 03 F5 03 00 31 00 2B  CD26
24D0: 01 F5 02 F0 3E 00 C4 00 F5 02 00 41 00 C4 00 F5  33DB
24E0: 03 00 4F 00 37 01 F5 03 00 56 00 35 01 F5 03 00  3F06
24F0: 60 00 37 01 F5 03 00 65 00 35 01 F5 02 F0 6A 00  D87C
2500: 21 01 F5 02 00 6D 00 21 01 F5 02 F0 70 00 12 01  8512
2510: F5 02 00 73 00 12 01 F5 03 00 9C 00 39 01 F5 03  5F43
2520: 00 A1 00 3B 01 F5 03 00 A5 00 37 01 F5 03 00 AA  FF54  ▶
```

```
2530: 00 35 01 F5 03 00 B9 00 2B 01 F5 03 00 C0 00 29   74F4
2540: 01 F5 04 00 C8 00 2D 01 F5 03 00 E2 01 2B 01 F5   39EC
2550: 03 00 E8 01 37 01 F5 03 00 ED 01 35 01 F5 03 00   2D38
2560: FC 01 2B 01 F5 02 00 08 02 0E 02 F5 02 00 0E 02   A941
2570: 22 02 F5 02 00 10 02 23 02 F5 02 00 12 02 D4 03   6434
2580: F5 02 00 14 02 22 02 F5 02 00 16 02 22 02 F5 02   105B
2590: 00 18 02 22 02 F5 02 00 1A 02 22 02 F5 02 00 1C   1B88
25A0: 02 22 02 F5 02 00 1E 02 22 02 F5 02 00 20 02 22   F09C
25B0: 02 F5 02 00 2C 02 3D 01 F5 02 00 33 02 3F 01 F5   ECC6
25C0: 02 00 58 02 37 01 F5 02 00 5C 02 35 01 F5 02 00   6F16
25D0: 67 02 37 01 F5 02 00 6B 02 35 01 F5 02 F0 81 02   F7A5
25E0: 41 01 F5 02 00 84 02 41 01 F5 02 00 87 02 3F 01   C1C1
25F0: F5 02 00 8B 02 3D 01 F5 02 00 8F 02 37 01 F5 02   2E79
2600: 00 93 02 35 01 F5 02 00 A5 02 3F 01 F5 02 00 A9   E549
2610: 02 3D 01 F5 02 00 B3 02 43 01 F5 02 00 B7 02 41   E821
2620: 01 F5 02 00 C7 02 37 01 F5 02 00 CB 02 35 01 F5   31E8
2630: 02 00 D8 02 37 01 F5 02 00 DC 02 35 01 F5 02 F0   1306
2640: E7 02 45 01 F5 02 00 EA 02 45 01 F5 02 F0 01 03   C543
2650: 4D 01 F5 02 00 04 03 4D 01 F5 02 F0 1B 03 6C 01   950C
2660: F5 02 00 1E 03 6C 01 F5 02 F0 35 03 89 01 F5 02   FB25
2670: 00 38 03 89 01 F5 02 F0 4F 03 A2 01 F5 02 00 52   4DEA
2680: 03 A2 01 F5 02 F0 69 03 C1 01 F5 02 00 6C 03 C1   ECE2
2690: 01 F5 02 00 AA 03 37 01 F5 02 00 AE 03 35 01 F5   49B0
26A0: 02 00 B9 03 37 01 F5 02 00 BD 03 35 01 F5 03 00   60DB
26B0: CC 03 2B 01 F5 03 00 D0 03 29 01 F5 02 00 D8 03   D1C2
26C0: 29 01 F5 02 00 05 04 37 01 F5 02 00 09 04 35 01   0B9C
26D0: F5 02 00 14 04 37 01 F5 02 00 18 04 35 01 F5 02   9787
26E0: 00 23 04 29 01 F5 02 00 63 04 29 01 00 00 00 00   4BD9
26F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   0000
2700: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   0000
2710: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   0000
2720: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   0000
2730: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   0000
2740: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   0000
```

COMPLETEZ AVEC DES 0 JUSQU'A 27FF

```
168  MACRO
169 &lab _SetSysWindow
170 &lab ldx £$4B0E
171  jsl $E10000
172  MEND
173  MACRO
174 &lab _SetPort
175 &lab ldx £$1B04
176  jsl $E10000
177  MEND
178  MACRO
179 &lab _GetPort
180 &lab ldx £$1C04
181  jsl $E10000
182  MEND
183  MACRO
184 &lab _MoveTo
185 &lab ldx £$3A04
186  jsl $E10000
187  MEND
188  MACRO
189 &lab _LineTo
190 &lab ldx £$3C04
191  jsl $E10000
192  MEND
193  MACRO
194 &lab _EraseRect
195 &lab ldx £$5504
196  jsl $E10000
197  MEND
198  MACRO
199 &lab _DrawCString
200 &lab ldx £$A604
201  jsl $E10000
202  MEND
```

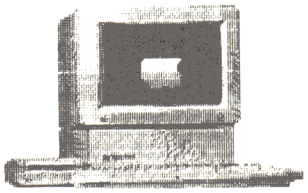
Si vous tapez les codes à partir du moniteur, procédez de la manière suivante pour sauver la routine ***ACCENT*** *:*

① ***CREATE ACCENT,T$B8***

② ***BSAVE ACCENT,T$B8,A$2000,L2048***

Table de conversion pour l'accessoire de bureau "Minuscules accentuées"

par Stephan HADINGER.



@/Ä	A/Å	B/Ç	C/É	D/Ñ	E/Ö	F/Ü
G/á	H/à	I/â	J/ä	K/ã	L/å	M/ç
N/é	O/è	P/ê	Q/ë	R/í	S/ì	T/î
U/ï	V/ñ	W/ó	X/ò	Y/ô	Z/ö	[/õ
\/ú	]/ù	^/û	_/ü			

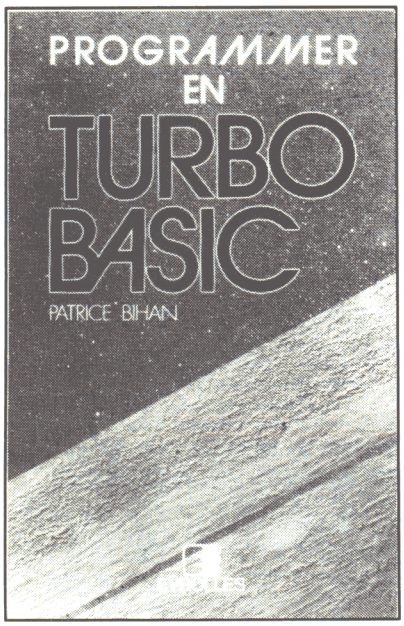
`/†	a/°	b/¢	c/£	d/§	e/•	f/¶
g/ß	h/®	i/©	j/™	k/´	l/¨	m/≠
n/Æ	o/Ø	p/∞	q/±	r/≤	s/≥	t/¥
u/µ	v/∂	w/∑	x/∏	y/π	z/ʃ	{/ª
|/º	}/Ω	~/æ	del/ø			

space/	!/·	"/‚	#/„	$/‰	%/Â	&/Ê
'/Á	(/Ë	)/È	*/Í	+/Î	,/Ï	-/Ì
./Ó	//Ô	0/	1/Ò	2/Ú	3/Û	4/Ù
5/ı	6/ˆ	7/˜	8/¯	9/˘	./˙	;/˚
</¸	=/˝	>/˛	?/ˇ			

caractères précédés de la touche 'ctrl' :

@/¿	A/¡	B/¬	C/√	D/ƒ	E/≈	F/Δ
G/«	H/»	I/…	J/	K/À	L/Ã	M/Õ
N/Œ	O/œ	P/–	Q/—	R/“	S/”	T/‘
U/’	V/÷	W/◊	X/ÿ	Y/Ÿ	Z/⁄	esc/¤
\/‹	]/›	^/fi	_/fl			

Votre bibliothèque INFORMATIQUE par NESTOR



• PROGRAMMER EN TURBO BASIC (par Patrice BIHAN)

Turbo Basic (un produit BORLAND) est un système de développement complet puisqu'il inclut à la fois un éditeur, un système de recherches d'erreurs et un compilateur. De plus, il est compatible avec les versions Basic IBM et Microsoft, Basica et GW.Basic. C'est actuellement la meilleure solution pour programmer efficacement en Basic sur IBM PC ou compatible. Autre avantage, ***Turbo Basic*** possède la rigueur d'un langage structuré et constitue une étape intéressante vers l'apprentissage de langages de plus haut niveau : Pascal et **C** entre autres.

Dans son livre, Patrice BIHAN vous propose une initiation rapide, mais apparemment efficace au ***Turbo Basic***.

Des exemples significatifs (une soixantaine) rendent ses exposés encore plus clairs. Un bon guide pratique.

(240 pages — Prix 195 F TTC)
***EYROLLES**, 61 bd St-Germain, 75005 PARIS*

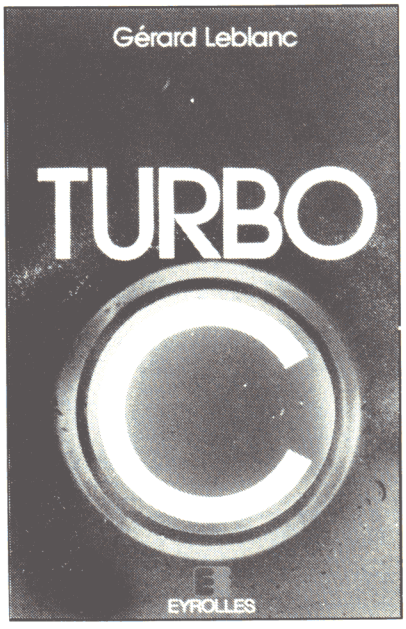
• TURBO C (par Gérard LEBLANC)

On sait déjà tout le bien que je pense de ***TURBO C*** de Borland (Ah ! si seulement nous avions l'équivalent sur Apple *IIGS !*). Franchement, s'essayer à programmer en ***Turbo C***, sur un IBM PC ou compatible, est un plaisir ! On n'en finit pas de s'émerveiller sur la rapidité du produit de Borland. Résultat : les éditeurs se déchaînent pour éclairer nos lanternes. Bravo !

Gérard LEBLANC a choisi l'explication par l'exemple. L'autodidacte de l'informatique que je suis approuve. Il a même envie de ne rien ajouter si ce n'est que ***TURBO C*** est un guide utile, voire nécessaire. Je ne lui adresserai qu'un reproche : l'absence d'une disquette contenant les exemples. On peut certes les taper, ce qui en facilite la compréhension, mais dans le monde moderne, chaque minute compte.

Bien sûr, cela n'enlève rien au livre, bien au contraire, et je connais des tas d'utilisateurs qui vont découvrir, en compagnie de Gérard LEBLANC, les mystères du **C** !

(352 pages — Prix 180 F) — ***EYROLLES***



• TURBO PASCAL SUR IBM PC ET AMSTRAD PC (par P.-M. BEAUFILS et N. LUTHER)

Je ne connais pas ***TURBO PASCAL*** sur IBM PC et je le regrette, mais j'ai déjà eu l'occasion de dire tout le bien que je pense de la version MACINTOSH de ce produit Borland.

Sur le champion d'Apple, nous disposons maintenant d'une version française de la documentation, ce qui rend l'apprentissage plus facile. J'ignore s'il en va de même sur PC.

Le guide de j'ai entre les mains devrait en tout cas simplifier la vie de nombreux programmeurs néophytes. C'est là encore une initiation par l'exemple et j'aime cela. Jeux, utilitaires, applications scientifiques, fichiers : tout y passe. On vous propose même une véritable interface de souris réalisée par logiciel !

Il y a aussi, dans le domaine du graphisme, des choses fort intéressantes.

Je suis certain que les lecteurs de ce ***TURBO PASCAL***, déploreront, comme moi, l'absence d'une disquette (décidément, je me répète, mais ce n'est pas encore du gâtisme !).

***EYROLLES**, 61 bd St-Germain, 75005 PARIS*

Emile SCHWARZ

Programmation comparée en GS Basic... et en C

GS Basic

J'ai défini cinq procédures : ***Initialisation***, ***Affiche.Présentation***, ***Menu.Affiche***, ***CelciusToFahrenheit*** et ***FahrenheiToCelcius***, pour montrer comment utiliser des procédures. Le *IIGS BASIC* traite plus rapidement les procédures que les sous-programmes avec accès GOSUB... RETURN. Je dois également préciser que le nom d'une procédure peut être en majuscules ou bien en minuscules. Locate 12,31 est "équivalent" à VTAB 12 : HTAB 31, en plus concis. L'affichage par une variable contenant 80 signes '=' est beaucoup plus rapide que l'utilisation d'une boucle. J'ai employé deux labels (DEBUT et FIN) avec des noms évocateurs également à des fins de démonstration.

```
 10 TEXT:HOME:GOTO DEBUT
 20 REM                          Conversion Celcius/Fahrenheit
 30 REM  (c) 26 Octobre 1987, E. SCHWARZ & TREMPLIN MICRO
 40 DEF PROC Initialise
 50 CtF$="Conversion Celcius en Fahrenheit"
 60 FtC$="Conversion Fahrenheit en Celcius"
 70 NdC$="Nombre en degrés Celcius: "
 80 NdF$="Nombre en degrés Fahrenheit: "
 90 PressRet$="Pressez <RETURN> SVP"
100 Ligne$="============================================================
    ========================"
110 END PROC Initialise
120 DEF PROC AffichePres
130 PRINT Ligne$;
140 PRINT "I" SPC(78)"I";
150 PRINT "I" SPC(23)FtC$ SPC(23)"I";
160 PRINT "I" SPC(78)"I";
170 PRINT "I" SPC(23)CtF$ SPC(23)"I";
180 PRINT "I" SPC(78)"I";
190 PRINT "I" SPC(30)"* 26 Octobre 1987 *" SPC(29)"I";
200 PRINT "I" SPC(78)"I";
210 PRINT "I" SPC(22)"(c) Emile SCHWARZ & TREMPLIN MICRO" SPC(22)"I
    ";
220 PRINT "I" SPC(78)"I";
230 PRINT Ligne$;
240 END PROC AffichePres
250 DEF PROC AfficheMenu
260 HOME:LOCATE 15,20:PRINT "1 - "CtF$
270 LOCATE 17,20:PRINT "2 - "FtC$
280 LOCATE 19,20:PRINT "3 - Quitter"
```

(suite page 56)

```
290 LOCATE 21,20:PRINT "Votre choix: ";
300 INPUT "";Choix
310 ON Choix<1 OR Choix>3 GOTO 290
320 END PROC AfficheMenu
330 DEF PROC CelToFahr
340 HOME:LOCATE 15,20:PRINT CtF$
350 LOCATE 17,20:PRINT NdC$;:INPUT "";Celcius
360 IF Celcius<=-273.11 THEN LOCATE 19,18:PRINT "> Zéro absolu =  -
    273.11° Celcius <":GOTO 380
370 LOCATE 19,20:PRINT NdF$;(Celcius/(5.0/9.0))+32
380 LOCATE 21,20:PRINT PressRet$;:INPUT "";A$
390 END PROC CelToFahr
400 DEF PROC FahrToCel
410 HOME:LOCATE 15,20:PRINT FtC$
420 LOCATE 17,20:PRINT NdF$;:INPUT "";Fahrenheit
430 IF Fahrenheit<=-459.6 THEN LOCATE 19,18:PRINT "> Zéro absolu =
    -459.6° Fahrenheit <":GOTO 450
440 LOCATE 19,20:PRINT NdC$;(5.0/9.0)*(Fahrenheit-32)
450 LOCATE 21,20:PRINT PressRet$;:INPUT "";A$
460 END PROC FahrToCel
470 DEBUT:PROC Initialise:PROC AffichePres:POKE 34,12
480 HOME:PROC AfficheMenu
481 DO
482 IF Choix=1 THEN PROC CelToFahr
485 IF Choix=2 THEN PROC FahrToCel
488 IF Choix=3 THEN FIN
490 UNTIL Choix<>3
500 GOTO 480:REM Boucle tant que l'option quitter n'a pas été sélec
    tionnée.
510 FIN:TEXT:HOME:LOCATE 12,31
520 PRINT "That's all folks!":LOCATE 22,1:END
```

Commentaire du programme en C

CELTOFAHR.C utilise diverses possibilités du langage **C** .

£define Lors de la compilation, le pré-processeur remplace le nom qui suit le *£define* par l'expression indiquée. Ainsi LIGNE sera remplacé par la ligne de '='.

J'ai défini quelque caractères spéciaux commentés dans le source : HOME, EOL, BELL et CR.

puts (put string) Envoie une chaîne de caractères à l'écran. Cette chaîne sera affichée beaucoup plus rapidement par *puts* que par *printf*.

while(d != 1) A été utilisé comme boucle d'attente. Il faut que la variable *d* soit égale à *1* pour sortir de la boucle. Pour quitter le programme, il y a la même boucle, mais avec une variable et une valeur différente.

printf("çt%19 %sçn", CHAINE1) Affiche la chaîne 1 *(%s)* en HTAB 19 *(çt%19)*.

Le reste du programme ne présente aucune difficulté particulière.

```
/*=============================================================================

                        Conversion Fahrenheit/Celcius

                              Emile SCHWARZ

                             20 Octobre 1987

=============================================================================*/

#include <stdio.h>

#define  LIGNE     "============================================================
#define  LIGNE1    "I                                          Cette ligne compte normalement
#define  CHAINE0   "Conversion Fahrenheit en Celcius"     80 fois le signe =
#define  CHAINE1   "Conversion Celcius en Fahrenheit"
#define  CHAINE2   "       Par Emile SCHWARZ        "
#define  CHAINE3   "         Octobre 1987           "
#define  CHAINE4   "(c) E. SCHWARZ et TREMPLIN MICRO"
#define  CHAINED   "I                          "
#define  CHAINEF   "                        I"
#define  PRESSNO   "              > Pressez 1 pour continuer <  "

#define  HOME     "çf"       /* Envoie un form feed à l'écran              */
#define  EOL      "çn"       /* Caractère de fin de ligne                  */
#define  BELL     "ç7"       /* Equivalent au CTRL-G du basic              */
#define  CR       "çr"       /* Retour Charriot                            */

/*=================== En premier il faut tracer un cadre ==================*/

AffichePres()
é
   puts(HOME);                  /* Efface l'écran                          */
   puts(LIGNE, EOL);            /* Trace un cadre                          */
   puts(LIGNE1, EOL);
   printf("%s%s%sçn", CHAINED, CHAINE0, CHAINEF); /* et ce qu'il faut      */
   puts(LIGNE1, EOL);
   printf("%s%s%sçn", CHAINED, CHAINE1, CHAINEF); /*    dedans             */
   puts(LIGNE1, EOL);
   printf("%s%s%sçn", CHAINED, CHAINE2, CHAINEF);
   puts(LIGNE1, EOL);
   printf("%s%s%sçn", CHAINED, CHAINE3, CHAINEF);
   puts(LIGNE1, EOL);
   printf("%s%s%sçn", CHAINED, CHAINE4, CHAINEF);
   puts(LIGNE1, EOL);
   puts(LIGNE, EOL, CR);
è

/*================================== Menu ==================================*/

AfficheMenu()
é
   printf("çnçt%20 1 - %sçnçn", CHAINE0);
   printf("çt%20 2 - %sçnçn", CHAINE1);
   printf("çt%20 3 - Quitterçnçn");
```

(suite page 58)

```
   printf("çt%22 - Votre choix: ");
è

/*============================ Celcius en Fahrenheit ======================*/

CelFahr()
é
float  Fahrenheit,Celcius;
int    d;
AffichePres();                  /* Efface l'écran & inscrit la présentation */
printf("çnçnçt%19 %sçnçn", CHAINE0);
printf("çt%19 Nombre en Celcius: ");
scanf("%f", &Celcius);
   if (Celcius <= -273.11)       /* Dans ce cas, il y a une erreur!          */
       é
       puts(BELL);              /* Ring the bell!                           */
       printf("çnçt%19 > Zéro absolu à -273.11° Celcius <çn");
       è
   else                         /* Sinon, on transforme.                    */
       é
       Fahrenheit = (Celcius / (5.0 / 9.0)) + 32;
       printf("çnçt%19 Valeur Fahrenheit: ");
       printf("%3.2fçnçn", Fahrenheit);
       puts(PRESSNO);
       while(d != 1)            /* Il faut taper '1' pour continuer         */
           é
           scanf("%d", &d);
           è
       è
è

/*============================ Fahrenheit en Celcius ======================*/

FahrCel()
é
float  Fahrenheit,Celcius;
int    d;

AffichePres();                  /* Efface l'écran & inscrit la présentation */
printf("çnçnçt%19 %sçnçn", CHAINE1);
printf("çt%19 Nombre en Fahrenheit: ");
scanf("%f", &Fahrenheit);
   if (Fahrenheit <= -459.6)    /* Dans ce cas, il y a une erreur!          */
       é
       puts(BELL);              /* Ring the Bell!                           */
       printf("çnçt%19 > Zéro absolu à -459.6° Fahrenheit <çn");
       è
   else                         /* Sinon, on transforme.                    */
       é
       Celcius = (5.0 / 9.0) * (Fahrenheit - 32.0);
       printf("çnçt%19 Valeur Celcius: ");
       printf("%3.2fçn", Celcius);
       puts(PRESSNO);
       while(d != 1)            /* Il faut taper '1' pour continuer         */
```

```
            é
            scanf("%d", &d);
            è
        è
è


/*============================ Programme principal ======================*/

main()
é
int     choix;                  /* Déclare la variable                  */

   do                           /* Exécution tant que le choix est      */
       é                        /*    différent de 3                    */
       AffichePres();
       AfficheMenu();
       scanf("%d",&choix);      /* En basic: INPUT "";choix             */
       if (choix == 1)
              CelFahr();
       if (choix == 2)
              FahrCel();
       if (choix < 1 ùù choix > 3 )
              choix = 0;
       è
   while (choix != 3);          /* Tant que 'choix' est différent de 3 -> do*/
   puts(HOME);                  /*          Envoie  un Home             */
   puts(BELL);                  /*          1 bip et c'est terminé...   */
   puts("Bye bye...");
è
```

• Amusette

Simplement pour vous amuser, comparez tout de même l'affichage de ce titre, en LM, avec celui que vous obtenez en **C** ... **Nestor.**

```
*801L
1=m    1=x    1=LCbank (0/1)

00/0801: A9 03        LDA £03
00/0803: 20 95 FC     JSR FC95
00/0806: 20 58 FC     JSR FC58
00/0809: A0 00        LDY £00
00/080B: B9 29 08     LDA 0829.Y
00/080E: C9 80        CMP £80
00/0810: 90 06        BCC 0818 é+06è
00/0812: 20 ED FD     JSR FDED
00/0815: C8           INY
00/0816: 80 F3        BRA 080B é-0Dè
00/0818: C9 00        CMP £00
00/081A: F0 0C        BEQ 0828 é+0Cè
00/081C: AA           TAX
00/081D: B9 28 08     LDA 0828.Y
00/0820: 20 ED FD     JSR FDED
00/0823: CA           DEX
00/0824: D0 FA        BNE 0820 é-06è
00/0826: 80 ED        BRA 0815 é-13è
00/0828: 60           RTS

*829.87D
00/0829:AA 50 A0 4C A0 AA AA-*P L **
00/0830:A0 04 C3 CF CE D6 C5 D2 D3 C9 CF CE A0 C4 C5 A0- .CONVERSION DE
00/0840:C4 C5 C7 D2 C5 D3 A0 C6 C1 C8 D2 C5 CE C8 C5 C9-DEGRES FAHRENHEI
00/0850:D4 A0 C5 CE A0 C3 C5 CC C3 C9 D5 D3 A0 CF D5 A0-T EN CELCIUS OU
00/0860:C3 C5 CC C3 C9 D5 D3 A0 C5 CE A0 C6 C1 C8 D2 C5-CELCIUS EN FAHRE
00/0870:CE C8 C5 C9 D4 A0 05 AA AA A0 4D AA 50 00-NHEIT .** M*P.
```

SPRINT

le traitement de texte de Borland

CHAPEAU ! *Une demi-heure. C'est le temps qu'il m'a fallu pour sortir mes trois premières lignes de texte sur une imprimante Epson LQ 2500, à partir d'un compatible Donatex. J'avoue à ma grande honte avoir "gaspillé" une bonne vingtaine de minutes pour paramétrer ma sortie imprimante, mais je suis complètement nul : c'était d'une simplicité tellement déconcertante que je préfère ne point m'étendre davantage sur ce problème "personnel". Je vous dois un autre aveu : en général, c'est seulement après avoir tâtonné un bon bout de temps que je me décide à prendre connaissance de la documentation dont, à l'image de la majorité des utilisateurs, je déplore la pauvreté, mais que je ne lis pratiquement jamais. Un exemple à ne pas suivre. Evitez de laisser traîner cette page dans une pièce fréquentée par vos enfants !*

• J'ai tout compris !

Mais revenons à SPRINT. Je n'étais pas spécialement bien disposé à son égard. On en avait écrit tant de bien dans les pages publicitaires insérées un peu partout que j'étais plutôt décidé à en déceler les éventuels défauts que les probables qualités : quand on s'appelle Borland, on ne peut pas se permettre de miser sur un mauvais produit.

Au moment où je rédige ces lignes, je suis loin d'avoir testé toutes les ressources avouées de SPRINT. Je me suis néanmoins astreint à sortir une bonne douzaine de textes d'une vingtaine de lignes et à les mémoriser pour la postérité. D'abord avec prudence : comme je dispose évidemment d'un disque dur (c'est tellement bon marché, sur Donatec, qu'il n'y a aucune raison de s'en priver), je ne tenais pas à en compromettre l'irremplaçable contenu par quelque manœuvre maladroite. Avec moi, notamment sur un PC, il faut s'attendre à tout. Je me sens tout de même plus à l'aise devant mon GS ou devant un Macintosh. Normal, non ? Mais je n'ai rien saboté. Et j'ai tout compris... ou presque.

Evidemment, au début, j'usais de la multitude de fenêtres que Borland ouvre en cascade, mais j'ai vite réalisé (il m'arrive aussi d'ouvrir et de parcourir la doc... quand ça m'arrange !) que la plupart des manœuvres pouvaient être simplifiées. Tenez ! un simple CTRL-F10 lance l'impression du texte en cours. J'ai abondamment mélangé les styles de caractères : normal, souligné, gras, italique... je vous promets que si mon imprimante avait pu parler elle m'aurait assurément traité de "machiavélique emm...", et elle aurait eu raison.

• Convivialité et efficacité

Première constatation : si l'on peut parler de convivialité quelque part, c'est bien chez Borland. Tenez ! un simple détail : la sauvegarde du texte est automatique. C'est judicieux quand on destine le produit à des étourdis comme moi, mais c'est utile pour tout le monde : qui n'a jamais oublié de sauvegarder une prose qui aurait indiscutablement fait date dans l'histoire de la littérature française ? Vous ? Vous aurez une médaille : l'effigie de Jean-Louis Gassée rêvant à un Macintosh-IBM !

Deuxième consttation... je vous y prends ! Vous n'aviez pas remarqué la faute de frappe qui afflige le deuxième mot du présent alinéa ? Avec Sprint, ça ne passe pas !

Un signal sonore vous rappelle à l'ordre. Je l'ai entendu... non sans surprise car n'ayant rien fait pour cela, je ne m'attendais pas à bénéficier de la correction automatique. Inutile de préciser que je me suis ensuite amusé à taper des tas de petites bêtises. La correction en direct des mots n'est certes pas parfaite (ne vous

attendez pas à ce que l'on vous signale des erreurs complexes de syntaxe), mais elle se révèle terriblement efficace.

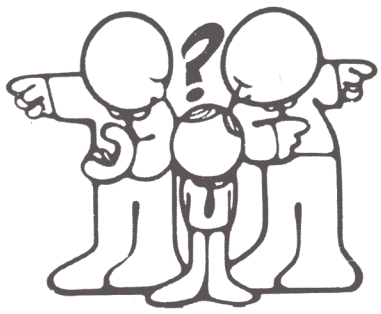
Je n'ai pas eu le temps de travailler avec une autre interface que celle — tout à fait standard — qui m'a permis ces brillants débuts. Sachez que vous pouvez retrouver sur Sprint toutes les commandes auxquelles vous êtes habitués sur un autre logiciel. Quelle simplification dans l'apprentissage !

Vous pouvez aussi créer votre interface personnelle, grâce au Kit du programmeur Sprint (ne l'ayant pas eu entre les mains, je ne vous en parlerai pas, mais il doit en tout cas être acheté séparément). Je suppose qu'il est indispensable de passer par lui pour que Sprint devienne réellement compatible avec les photocomposeuses, mais l'information a pu échapper à mon attention.

• Conclusion

Je suis réellement emballé par Sprint, traitement de texte original, associant excellemment convivialité, simplicité et efficacité. Un futur best-seller de l'informatique. Chapeau Borland !

Guy-Hachette.



Quelques exemples en GS Basic

PRINT USING existe...

... vous le rencontrerez !

```
10 HOME
20 FOR I=1 TO 10
30   PRINT USING "£££.£££££";100*RND(3);
40   A=100*RND(9):HPOS=15:PRINT USING "££.££££££";A;
50   HPOS=30:PRINT INT(A);:HPOS=38:PRINT USING "£££";A
60 NEXT
```

Deux utilisations de TIME

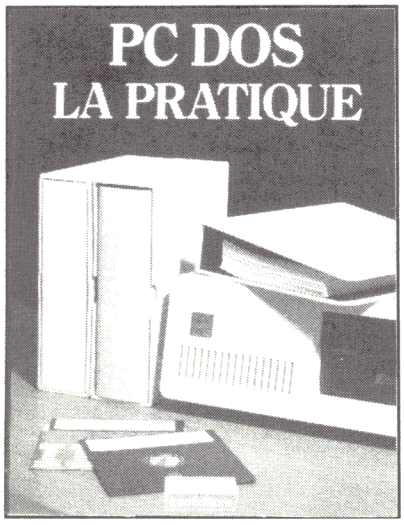
```
10 HOME
20 LOCATE 24,37
30 FOR I=1 TO 3
40   A=TIME(0)
50   PRINT TIME(I);" ";
60 NEXT
80 ON KBD  GOTO 100
90 GOTO 20
100 ON KBD  GOTO 100
110 LOCATE 0,0
```

```
10 HOME
20 LOCATE 24,37
40 PRINT TIME$;
80 ON KBD  GOTO 100
90 GOTO 20
100 ON KBD  GOTO 100
110 LOCATE 0,0
```

COPIE DE FICHIER TEXTE

```
10 HOME:INVERSE
20 PRINT "COPIE D'UN FICHIER TEXTE"
30 NORMAL:LOCATE 10,1
40 INPUT "NOM DU FICHIER ";A$
50 OPEN A$, FILTYP= TXT  FOR INPUT AS£10
60 INPUT "TITRE DE LA COPIE ";A$
70 OPEN A$, FILTYP= TXT  FOR OUTPUT AS£11
80 ON EOF£10 PRINT "DONE":CLOSE:END
90 INPUT£10;A$:PRINT£11;A$:GOTO 90
```

Votre bibliothèque INFORMATIQUE



• PC DOS — LA PRATIQUE
(par Chris DEVONEY)

QUE, la nouvelle collection d'Inter-Editions est réellement une réussite. A quand, dans cette série, le pendant, pour Apple *IIGS*, de ce PC DOS ?

Il faut dire que la méthode adoptée par l'auteur (traduit par Bertrand Blumereau) présente de nombreux avantages. Des exposés clairs, documentés quand le sujet l'exige, mais jamais d'une manière oiseuse.

Dans la partie consacrée au manuel de référence des commandes du DOS, chaque mot-clé est expliqué suivant un schéma particulièrement efficace : fonction, syntaxe, paramètres, règles, exemples éventuels, remarques et messages. Pour certaines fonctions, cela se traduit par une page de texte, mais pour d'autres par des développements beaucoup plus longs.

PC DOS — LA PRATIQUE répondra à l'attente de tous les utilisateurs, mais surtout à celle des gens qui, comme moi, tapotent plus souvent sur le clavier d'un Apple que sur celui de quelque "clone" plus ou moins rigolo.

On disposera, avec cet ouvrage, non seulement d'un aide-mémoire (voir l'index), mais d'un guide sûr, conçu par un auteur qualifié. **Nestor.**

***InterEditions**, 87 avenue du Maine 75014 PARIS (584 pages sous reliure souple).*

• INITIATION À L'ALGORITHMIQUE ET AUX STRUCTURES DE DONNÉES
(par J. COURTIN et I. KOWARSKI)
Récursivité et structures de données avancées.

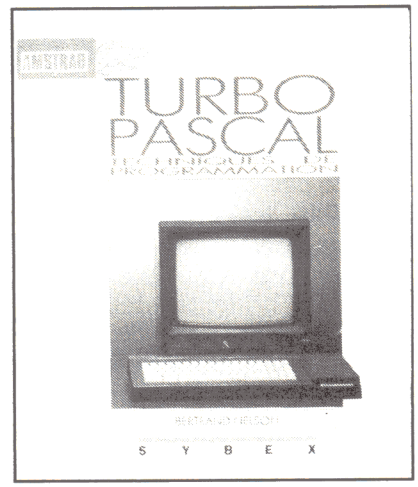
Au début de 1987, est paru le premier volume de cette ***Initiation à l'algorithmique et aux structures de données*** consacré à la programmation structurée et aux structures de données élémentaires.

Dans ce deuxième volume qui est la suite logique du premier, Jacques Courtin et Irène Kowarski abordent la récursivité et les structures de données avancées. Cet ouvrage utilise les mêmes méthodes et les mêmes notations que le premier. Il est consacré principalement à la programmation récursive et à la construction d'algorithmes classiques et fondamentaux sur des structures de données plus avancées que sont les listes chaînées, les piles, les files d'attente, les tables et les arbres. Il constitue ainsi une bonne préparation à l'utilisation des langages de l'intelligence artificielle (Lisp, Prolog) et aux méthodes modernes d'analyse des systèmes d'information. Cet ouvrage s'adresse aux étudiants des spécialités informatiques (I.U.T., B.T.S., D.P.C.T., D.E.S.T., LICENCE et MAÎTRISE, M.S.T., M.I.A.G.E., ÉCOLES D'INGÉNIEURS, INSTITUT DE PROGRAMMATION, D.E.S.S.,...) et aux lecteurs souhaitant acquérir les bases solides nécessaires à une bonne activité de programmation quel que soit le langage support envisagé.

Les algorithmes sont écrits dans un langage proche du Pascal et leur traduction dans un langage classique (Pascal, Fortran, Cobol, Basic...) est aisée. De très nombreux exemples et exercices, dont certains sont corrigés, permettent de se familiariser avec la démarche proposée et de vérifier que la méthode a bien été assimilée.

(Coll. "Dunod Informatique", Dunod 1987, 15,5×24, 360 pages, broché)

***DUNOD**, 17 rue Rémy-Dumoncel 75661 PARIS Cedex 14*



• TECHNIQUES DE PROGRAMMATION EN TURBO PASCAL SUR AMSTRAD
(par Bertrand NELSON)

Cet ouvrage se propose de fournir aux utilisateurs déjà familiarisés au langage Assembleur, toutes les informations utiles pour exploiter le logiciel interne (Firmware) de l'Amstrad aussi complètement que possible.

Le livre présente, dans un premier temps, les outils de base qui permettent une programmation avancée en Turbo Pascal, le tout étant illustré de nombreux exemples. Il s'attarde, dans un deuxième temps, sur les problèmes de comptabilité entre les deux systèmes CPM tournant sur Amstrad avec toutes les informations pour l'adaptation des programmes. L'auteur fournit tout au long de son ouvrage, une multitude de petites procédures utilitaires.

La deuxième partie de l'ouvrage comporte de nombreux programmes plus conséquents, essentiellement en Assembleur mais aussi en Turbo Pascal standard.

Cet ouvrage se destine surtout aux programmeurs désireux de pouvoir travailler en Pascal avec les mêmes possibilités qu'avec le Basic et l'Assembleur.

L'auteur : Bertrand NELSON est physicien-chimiste. Il est déjà l'auteur d'un logiciel de comptabilité générale en Turbo Pascal sur Amstrad et adaptable sur MS-DOS.

Un livre broché de 270 pages au format 190×230.

***SYBEX**, 6-8 impasse du Curé, 75018 PARIS*

COURRIER

LA QUESTION :

Je m'adresse à vous, car je ne suis pas parvenu à implanter dans le STARTUP d'un programme d'application, à partir de l'UTILITAIRE SYSTEME de mon IIc, un programme de configuration du port série n°1.

Mon logiciel fonctionne très bien, mais l'impression des tableaux nécessite de supprimer le LF après 80 caractères de la configuration de base ; on doit à chaque fois recharger le NIP modifié à partir de l'Utilitaire.

J'ai tenté d'implanter l'utilitaire de Claude AUBRY dans le but d'utiliser les copies d'écran par Ampersand Hard. Il convient de noter que cet utilitaire plaisant ne fonctionne, sur un IIe, qu'avec une carte parallèle Apple à l'exclusion de toute autre interface (Grappler ne fonctionne pas mieux qu'Apple Work sur mon IIc et ma FX80).

Si vous pouviez m'indiquer comment implanter une routine de configuration des ports séries sur un logiciel, et ce sans que son fonctionnement soit perturbé sur un IIe avec carte parallèle, vous auriez droit à ma plus profonde gratitude !

J.-M. S. (76170 LILLEBONNE)

LA RÉPONSE :

Tremplin Micro a déjà répondu à votre question concernant la configuration du port série du *IIc* (Luc Moreau dans *T.M. n°9* et Yvan Kœnig dans *T.M. n°13* page 73). Je profite de l'occasion pour rappeler quelques évidences.

Le *IIc* stocke dans 4 trous d'écran de sa mémoire auxiliaire le configuration du port série n°1. Ces informations sont utilisées chaque fois que l'on fait un PR£1. Pour les modifier, il est possible d'employer les utilitaires système mais ce n'est pas très pratique. Ces informations sont les suivantes :

$478 aux — Contient par défaut £%10011110 ce qui signifie 8 bits de data, 2 bits de stop, 9600 bauds.
- bit0 à 3 : Vitesse de transmission en bauds (cf. table).
- bit4 : Toujours 1.
- bit5 à 7 : Format info (cf. table).

$479 aux — Contient par défaut £%00001011 pas de bit de parité
- bit0 à 4 : Ne pas toucher.
- bit5 à 7 : Codage parité (cf. table).

$47A aux — Contient par défaut £%01000000 pas d'écho écran, LF après CR, mode imprimante.
La signification précise des bits est la suivante :

bit0	1 = communication	0 = imprimante
bit1 à 5	Toujours 0	
bit6	1 = LF après CR	0 = pas de LF
bit7	1 = écho écran	0 = pas d'écho

$47B aux — Contient par défaut £%01010000 c'est-à-dire tampon ligne de 80 caractères.

VITESSES en BAUDS

1	50	9	1800
2	75	10	2400
3	110	11	3600
4	135	12	4800
5	150	13	7200
6	300	14	9600
7	600	15	19200
8	1200		

FORMATS

	data	stop
0	8	1
1	7	1
2	6	1
3	5	1
4	8	2
5	7	2
6	6	2
7	5	2

PARITÉS

0	sans
1	impaire
2	sans
3	paire
4	sans
5	MARK
6	sans
7	SPACE

Pour régler votre problème, il me semble qu'il suffirait de modifier le tampon ligne, en le mettant, soit à 0, soit à une valeur supérieure à 80. C'est ce que je faisais dans *Tremplin Micro n°13* où j'installais 96. Je vous rappelle la routine : *(suite page 64)*

```
300 :  AD 18 C0      LDA  C018
303 :  48            PHA
304 :  8D 01 C0      STA  C001
307 :  A9 60         LDA  £$60
309 :  8D 55 C0      STA  C055
30C :  8D 7B 04      STA  47B
30F :  8D 54 C0      STA  C054
312 :  68            PLA
313 :  30 03         BMI  318
315 :  8D 00 C0      STA  C000
318 :  60            RTS
```

BSAVE SETLARGEUR,A$300,L$19

Pour votre second problème, il me semble que vous mélangez tout.

Les problèmes que l'on rencontre à l'impression peuvent avoir plusieurs origines :

- Un ou plusieurs bug(s) dans le programme (j'ai utilisé HardCopy, ce n'est pas le cas).
- Incompatibilité programme / interface. C'est le cas APPLEWORKS / ancienne interface parallèle EPSON (j'ai déjà signalé qu'EPSON assurait gratuitement l'échange contre la version £8133).
- Incompatibilité interface / imprimante (il y a des inconscients qui branchent n'importe quoi sur n'importe quoi).
- Incompatibilité programme / imprimante.

Dans votre cas, c'est ce qui se passe. Carte *//* APPLE (ou vieille carte EPSON et sans doute GRAPPLER *//*) + APPLE-DMP, ça fonctionne. Carte SSC, port série *IIc* (sans doute port série GS mais je n'ai pas essayé) + IMW 1 ou 2, ça marche. C'est normal, Claude AUBRY a écrit pour CES configurations. Si vous utilisez une imprimante EPSON ou similaire, ça ne fonctionnera pas et cela pour plusieurs raisons :

- Lorsque APPLE envoie un octet correspondant à une colonne de 8 points, le bit0 correspond au point HAUT alors que chez EPSON, il correspond au point BAS.
- Les chaînes de commande gérant les diverses fonctions des imprimantes sont spécifiques à chaque constructeur.
- Une imprimante APPLE peut imprimer, dans un caractère personnalisé, deux points consécutifs sur une même horizontale. EPSON ne le peut pas.

Si vous tenez à utiliser la recopie d'écran du bureau de MINIE sur EPSON, il faut vous plonger dans le source et tout adapter à la logique EPSON, mais vous aurez, à coup sûr, un résultat médiocre pour les inverses et les icônes. Il vous serait possible, sans trop de difficulté, de remplacer l'utilisation de caractères perso par une autre formule. Par exemple, vous pourriez recoder les contrôles en majuscules "grasses" et les icônes en majuscules indices. Pour ce faire, il vous faut disposer de MERLIN PRO ou vous adresser à Claude AUBRY qui me semble le mieux placé pour adapter MINIE & CO. **Y. KŒNIG.**

QUESTION :

J'ai lu dans le numéro 15 du 04/07/87 de TREMPLIN MICRO (bas de la colonne 1, page 61) — Un correcteur d'orthographe de 100 000 mots — Comment est-ce réalisé sur PC ?
Comment est-ce réalisable sur APPLE II, sinon pourquoi l'eau à la bouche ?

J'ai créé sur mon APPLE II+ équipé d'une carte langage et d'une ROM minuscule, un dictionnaire de mots croisés de 82 068 mots accentués, non conjugués, soit 712 051 caractères DATA frappés.*

J'ai besoin des 2 faces de 5 disquettes (programmes compilés, organisés et optimisés pour un minimum d'encombrement), et sans les programmes, les DATA occupent 8 disquettes !

Est-ce qu'un programme en langage machine utiliserait beaucoup moins de place et tournerait plus vite d'un programme BASIC COMPILE ? Pourrait-il signaler instantanément une faute d'orthographe portant sur des mots de longueur quelconque (ex. : invraisemblance ou anticonstitutionnellement) ou des verbes conjugués (ex. : seoir) ? **C. B. (02110 BOHAIN)**

** A qui pourrais-je m'adresser pour le publier ? Y-a-t-il des éditeurs de logiciels APPLE II qui seraient intéressés par un "dico" en français ?*

RÉPONSE :

Désolé ! mais il existe déjà, sur APPLE *IIe* un correcteur orthographique ANGLAIS de 114 000 mots. Il s'agit du *WEBSTER'S NEW WORLD SPELLER CHECKER*.
Editeur : SIMON & SCHUSTER (cf. NIBBLE mars 87).

Un programme en assembleur pur, bien écrit, tournerait probablement 2 fois plus vite que l'APPLESOFT compilé et serait environ moitié moins gourmand en espace mémoire (ce sont des ordres de grandeur).

Y. K.

NDLR : Il est tout à fait inutile, dans le cas de mots tels que COMPARAITRE, COMPASSION, COMPARUTION... de stocker les 5 premiers caractères ! Des tas d'astuces permettent des gains de place considérables.

QUESTION :

Dans le programme de la page 7 de Tremplin Micro n°15 vous recommandez 4 fichiers ainsi que dans le dossier system de contenir (STAR). Mais sur MOUSE DESK il n'y a pas celui de STAR ni de STARTUP et...

je ne sais où les trouver. Comme je vous commande la disquette n°15, je voudrais bien pouvoir faire tourner ce programme sur le GS. Je vous remercie à l'avance.

J. B. (94220 CHARENTON)

RÉPONSE :

Nicole semble avoir été trop exigeante. Je vous propose ci-dessous le contenu du directo principal et du DIR system de la disquette avec laquelle j'ai essayé PR.BASIC.

STARTUP est un programme très élaboré dont vous trouverez le listing ci-dessous (je n'ai pas cru nécessaire d'utiliser SIGNATURE).

1Ø PRINT CHR$ (4)"-PR.BASIC"

Y (24/07/87 12:45) VØØØ

Type	Blocs	Nom	Créé	Modifié	Long.
System	42	PRODOS	24/07/87 12:45	06/09/86 12:00	2Ø992
System	21	BASIC.SYSTEM	24/07/87 12:45	18/06/84 00:00	1Ø24Ø
*Direct	1	SYSTEM	24/07/87 12:46	24/07/87 12:50	512
AsfBAS	5	PR.BASIC	24/07/87 12:50	13/05/87 00:00	1618
Binary	1	PG.OBJ	24/07/87 12:50	13/04/87 10:22	246
AsfBAS	1	STARTUP	24/07/87 12:10	24/07/87 13:10	24

1183 Blocs disponibles sur 16ØØ

SYSTEM (24/07/87 12:46) VØØ4

Type	Blocs	Nom	Créé	Modifié	Long.
System	32	P8	24/07/87 12:46	06/09/86 12:00	15485
System	66	P16	24/07/87 12:46	06/09/86 12:00	33121
*Direct	1	SYSTEM.SETUP	24/07/87 12:46	24/07/87 12:48	512
*Direct	1	TOOLS	24/07/87 12:46	24/07/87 12:50	512
*Direct	1	DESK.ACCS	24/07/87 12:46	24/07/87 12:50	512
*Direct	1	FONTS	24/07/87 12:46	24/07/87 12:50	512

QUESTION :

J'ai tapé le programme "La boîte aux idées" paru dans Tremplin Micro n°8 et j'ai décidé de reprendre la routine "saisie du texte" dans un de mes programmes.

Je suis un programmeur novice (et je le regrette), pas pour longtemps j'espère, grâce à votre revue, et j'ai besoin de vous pour m'éclairer sur un certain point.

En utilisant la routine "saisie du texte" (l.19Ø à 37Ø) — comme il est prévu dans le programme ***Boîte*** *— je ne peux écrire que sur une ligne (78 caractères) ; or, je voudrais pouvoir écrire un texte sur une page entière (indépendamment du programme* ***Boîte****, bien sûr, mais dans un des miens, comme je l'ai déjà signalé).*

Pouvez-vous me donner les modifications à apporter pour que je puisse mettre fin à mes problèmes ? ***E. D. C. (14 ans) — (38000 GRENOBLE)***

RÉPONSE :

Ta demande est imprécise, cherches-tu :

- Une routine d'INPUT améliorée, forcément limitée à des chaînes de 239 caractères ?
- Une routine de saisie d'écran ?
- Un traitement de texte ?

Pour la routine d'INPUT vois, entre autres, NIBBLE mai 87 (OUTLINER).

Pour la saisie d'écran, *Tremplin Micro* a déjà traité le problème plusieurs fois.

Un traitement de texte n'est pas du ressort du courrier des lecteurs. APPLEWRITER ProDOS fonctionne très bien. Je te signale que VIF commercialise en Domaine Public : FreeWriter*, une version simplifiée d'APPLEWRITER que tu pourras décortiquer à loisir. Je te promets que tu y apprendras beaucoup de choses. Pour ce faire, je te conseille de te procurer l'assembleur MERLIN dont le désassembleur SOURCEROR est TRÈS commode. C'est avec lui que j'ai désassemblé, entre autres, DOS 3.3, ProDOS, BASIC.SYSTEM, APPLEWRITER dos puis ProDOS etc. **Y. K.**

** FREEWRITER, disquette 5", réf. 15019 : 60F — 3", réf. 65019 : 80F chez VIF/DP 5, rue Bassano, 75116 PARIS.*

QUESTION :

A propos de la couleur en DHGR (Apple IIc) pourquoi ECRIT DHGR (Tremplin Micro n°12) affiche-t-il en couleur "arc-en-ciel" et qu'après plusieurs RUN successifs (3 ou 4), finit-on par obtenir le BLANC souhaité ?
Ai-je mal lu votre revue ou bien cette question n'a-t-elle pas été abordée ?
D'avance merci. ***C. D. (81100 CASTRES)***

RÉPONSE :

Comme je l'ai déjà écrit dans *Tremplin Micro n°13* (page 72), sur le *IIc*, il faut activer le "switch" $CØ7E avant $CØ5E et DEMO.DHGR ne le faisait pas.

Ajoutez la ligne 39 POKE 49278,Ø et ça ira peut-être mieux... **Y. K.**

QUESTION :

Je suis depuis peu (10 jours), l'heureuse propriétaire d'un APPLE IIGS et, je l'avoue, j'en suis fière. Je commence à me régaler avec vos routines et "lui", il s'éclate.

Si je vous écris, c'est pour vous poser quelques questions (je vais abuser !) :

Existe-t-il plusieurs versions de l'unité centrale du GS ? (la mienne n'affiche pas MOUSE DESK en couleur, mais en noir sur fond blanc et la bordure dans la couleur que j'ai choisie par le tableau de bord).

Pourquoi DEMO/PAINT ne fonctionne-t-il pas ? (il se boot correctement, mais affiche après un écran blanc sans aucun menu déroulant ; il y a uniquement l'icône flèche). Il en va de même pour DELUXE/PAINT.

Par contre, GS/PAINT marche correctement. J'allais oublier... j'ai une amie qui a également acheté un GS, et MOUSE DESK est partiellement en couleur (lettre noire sur fond rose). Je vous fais de grosses bises...

Je finis par vous demander un conseil. Je me familiarise depuis peu à ProDOS. Je convertis certains programmes du bon vieux DOS 3.3 sur des disquettes 3" 1/2 en ProDOS, j'y ajoute BASIC-SYSTEM et le programme de mon choix pour le démarrage, un STARTUP et tout va très bien, mais certains fichiers T et Binaires refusent ProDOS.

Je n'ose pas vous demander pourquoi, vous risqueriez de penser que je ne lis plus votre revue, et pourtant, je suis fidèle et j'attends toujours les numéros de Tremplin Micro avec impatience et croyez-moi, c'est avec la même impatience que j'attends une réponse. ***Alexandra C. (69007 LYON)***

RÉPONSE :

Merci pour les grosses bises.

Je ne possède ni *DEMO.PAINT* ni *DELUXE.PAINT*. Etes-vous sûre que votre carte mémoire est assez garnie ? Je sais que la présentation GS exige, par exemple, 2 lecteurs 3"5 et une carte 1méga. *GS.PAINT* se plante lorsqu'on essaie de l'utiliser sur une machine dont la carte 1 méga est configurée avec un disque virtuel de plus de 500K. Ça devrait figurer dans les manuels, mais ça n'y est pas. Il me semble que *DEMO.PAINT* n'est pas un programme commercialisé, mais simplement un outil pour les concessionnaires, et si c'est bien le cas, rien d'étonnant à ce qu'il exige 1 méga. Votre concessionnaire devrait pouvoir être plus précis.

Ne vous leurrez pas, APPLE ne nous a fourni AUCUNE information particulière. Nous devons tout découvrir par expérience.

Pour vos problèmes de transfert DOS 3.3 vers ProDOS, je vois deux possibilités :

- Vous essayez de transférer des fichiers TXT à accès direct avec les utilitaires APPLE qui ne savent pas le faire. Le plus efficace serait de le faire avec COPY II PLUS.
- Pour les fichiers BIN, il est probable qu'il s'agit :
  - soit de programmes appelant directement DOS 3.3, ce que ProDOS aura du mal à comprendre ;
  - soit de programmes se plaçant juste endessous de DOS 3.3, ce qui est inacceptable sous ProDOS qui passe son temps à déplacer HIMEM au gré de l'ouverture et de la fermeture des fichiers.

Il serait plus simple d'acheter pour $29,95 + $2,50 de port, le programme UNIDOS de NIBBLE qui permet de travailler en DOS 3.3 sur des disques 3"5. Je dois préciser qu'UNIDOS étant un DOS 3.3 sérieusement remanié, bon nombre de patches plus ou moins classiques ne sauraient lui être appliqués.

Puisque vous avez un GS, je vous conseille une petite gymnastique. Formatez une disquette et copiez-y le fichier P8 qui se trouve dans un sous-catalogue de MOUSE.DESK. Sur la disquette ainsi garnie, faites RENAME P8,PRODOS et vous aurez ainsi un ProDOS 8 bits reconnaissant l'horloge du GS et beaucoup d'autres gâteries (des bugs entre autres). N'oubliez pas de compléter avec BASIC.SYSTEM, STARTUP et compagnie. **Y. K.**

NDLR : Nous avons lu dans les colonnes de notre excellent confrère américain Call APPLE qu'APPLE US reconnaît avoir livré les premières séries de GS avec un composant "défectueux". Il s'agit du VGC qui gère la vidéo. L'anomalie se traduit, entre autres choses, par un moirage et par un écran rosé alors qu'il devrait être gris. Apparemment, ils auraient réussi à régler le problème puisque votre écran est gris.

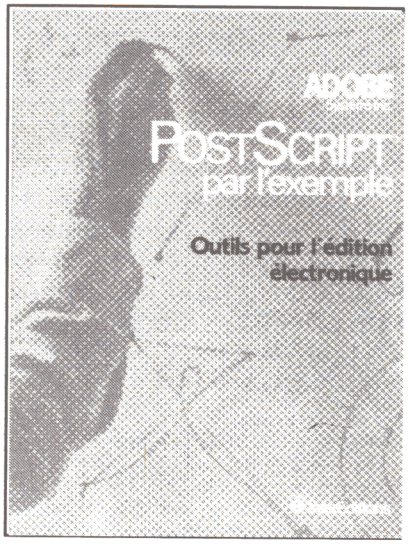
Cette réponse d'Yvan KŒNIG remonte à plusieurs mois. Au moment où nous bouclons ce numéro, il semble qu'Apple France se soit enfin décidé à modifier gratuitement tous les GS défectueux...

Nous ne répondons qu'aux lettres accompagnées d'une enveloppe affranchie pour le retour.

Si vous nous adressez un programme, n'oubliez pas d'y joindre la disquette.

Votre bibliothèque INFORMATIQUE

par NESTOR



• PostScript par l'exemple

Outils pour l'édition électronique

- Adobe Systems,Inc

On sait que ***PostScript*** est déjà implanté sur la LaserWriter d'Apple ainsi que sur les ordinateurs de la nouvelle gamme IBM.

PostScript est un langage de description de page permettant de décrire exactement, à l'intention de l'imprimante qui l'exécutera, l'aspect des textes, images et objets graphiques que l'on désire éditer. On peut dire que ***PostScript*** est rapidement devenu La référence en matière de PAO (Publication Assistée par Ordinateur). Sa souplesse et sa puissance sont maintenant reconnues... y compris par IBM, ce qui prouve d'ailleurs qu'Apple avait fait le bon choix. Cet ouvrage a été écrit par une équipe d'Adobe Systems, Inc, la Société qui a inventé le langage... ce qui explique sans doute sa clarté (le traducteur, Alain Kadé, n'y est pas non plus pour rien).

Dans une première partie, les auteurs nous proposent une visite guidée de ***PostScript***, une sorte de revue en détail d'ailleurs fort agréable. La seconde partie est bourrée d'exemples de difficulté croissante, bien commentés.

Un bon bouquin, sous une reliure souple particulièrement adaptée à ce genre de livre.

(260 pages, 232 F TTC)

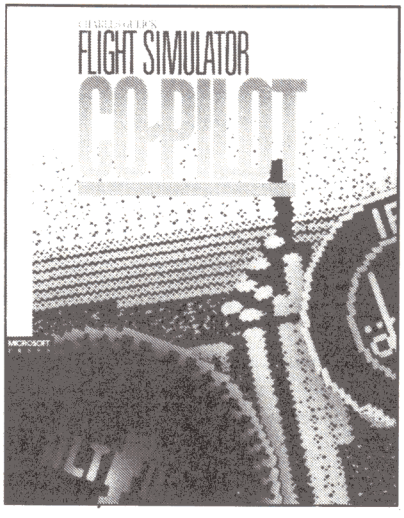
***InterEditions**, 87 avenue du Maine 75014 PARIS*

• FLIGHT SIMULATOR CO-PILOT

- Charles GULICK

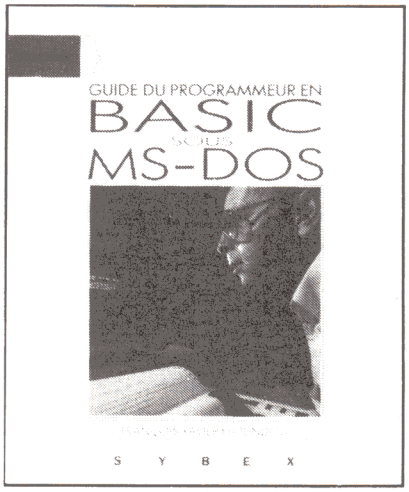
Qui ne connaît pas ce logiciel de simulation de vol, le plus vendu dans le monde ? Des milliers d'élèves-pilotes jouent avec dans tous les aéro-clubs de la planète !

Dans ce livre, Charles Gulick vous indique comment tirer la quintessence d'un logiciel qui n'a pas fini de nous étonner. Depuis les instructions de vol élémentaires jusqu'au vol de New York, en passant par les ascensions et descentes de base, les virages, la navigation avec et sans instru-ments... vous saurez tout sur ***FLIGHT SIMULATOR !***



(165 pages, 145 F TTC)

CEDIC/NATHAN (MICROSOFT PRESS)
***PSI**, 5, place du Colonel Fabien — 75491 PARIS CEDEX 10*



• GUIDE DU PROGRAMMEUR EN BASIC SOUS MS-DOS

- François-Xavier ELOUNDOU

Ce guide commence par des routines simples permettant de gérer l'écran, de créer un menu déroulant, de sélectionner et d'ouvrir des fenêtres, de faire de la saisie pleine page, etc.

La deuxième partie est consacrée à la gestion de fichiers, tandis que la troisième partie concerne des techniques aussi diverses que : sous-programmes en langage machine et niveaux de rupture.

Un aide-mémoire Basic complète ce guide. Il est lui aussi illustré de très nombreux exemples.

(685 pages — 248 F TTC)

***SYBEX**, 6-8 impasse du Curé, 75018 PARIS*

N°18

Bulletin de commande et d'abonnement

A retourner à : **TREMPLIN MICRO — Senillé — 86100 CHATELLERAULT**

Nom ..... Prénom .....

Adresse complète .....

Code postal |_|_|_|_|_| Ville .....

LES DISQUETTES FONCTIONNENT SOUS DOS ET ProDOS (à condition de posséder une version Apple de ce SYSTEME D'EXPLOITATION)

Qté.	DÉSIGNATION	PRIX UNITAIRE	TOTAL
	Abonnement à 6 numéros (un an) : FRANCE à partir du numéro : _____	190 F	
	Abonnement à 6 disquettes (un an) : FRANCE à partir du numéro : _____	600 F	
	MINIE LE BUREAU DE MINIE LES SOURCES DU BUREAU — Logiciels de Claude AUBRY	180 F 150 F 50 F	
	LIVRE : APPLE // PRODOS, GUIDE DU PROGRAMMEUR APPLESOFT	120 F	
	LIVRE : LE 65Ø2 PAS à PAS	120 F	
	LIVRE + DISQUETTE : CLINS D'ŒIL AU 65Ø2	160 F	
	LIVRE + DISQUETTE : ROUTINES LM POUR 65CØ2 ET 65Ø2	160 F	
	LIVRE + DISQUETTE : NOUVELLES ROUTINES POUR LE 65CØ2	160 F	
	LIVRE + DISQUETTE : PROGRAMMES LM 65CØ2 & 65C816	160 F	
	DISQUETTE DE JEU WAR	20 F	
	DISQUETTE DE JEU "Le Monde des Sorciers"	20 F	
	DISQUETTE N°18 + jeu "Le Monde des Sorciers"	105 F	
	DISQUETTE-INDEX (N°1 à 12)	30 F	
	SIGNATURE (CHECK-LIST)	30 F	
	RELIURE-ÉCRIN	40 F	
	DISQUETTE DE TREMPLIN MICRO : * *(IIe, IIc, GS)* ☐ N°1 ☐ N°4 ☐ N°7 ☐ N°10 ☐ N°13 ☐ N°16 ☐ N°2 ☐ N°5 ☐ N°8 ☐ N°11 ☐ N°14 ☐ N°17 ☐ N°3 ☐ N°6 ☐ N°9 ☐ N°12 ☐ N°15 ☐ N°18	105 F	
	NUMÉROS DE TREMPLIN MICRO : * ☐ N°1 ☐ N°2 ☐ N°3 ☐ N°4 ☐ N°5 ☐ N°6 ☐ N°7 ☐ N°8 ☐ N°9 ☐ N°10 ☐ N°11 ☐ N°12	30 F	
	☐ N°13 ☐ N°14 ☐ N°15 ☐ N°16 ☐ N°17 ☐ N°18	33 F	

* *Cochez la (ou les) case(s) de votre choix.*

Merci de libeller votre règlement à l'ordre de TREMPLIN MICRO / Editions JIBENA.

Participation aux frais d'envoi (gratuit pour les abonnés) + 10 F

Envoi en paquet-poste recommandé à partir de 400 F

Numéro d'abonné ou de client : _____ **Total à payer** _____ **F**

Mode de règlement choisi :

☐ Chèque

☐ Mandat-lettre

☐ Carte Bleue (complétez ci-contre)

VISA

N° de votre Carte Bleue |_|_|_|_| |_|_|_|_| |_|_|_|_| |_|_|_|_|

Date d'expiration |_|_|_|_| **Signature (obligatoire)**

Montant à débiter _____ **F**

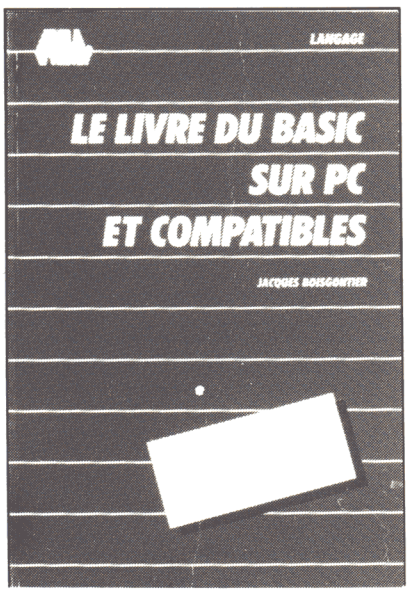
Votre bibliothèque INFORMATIQUE

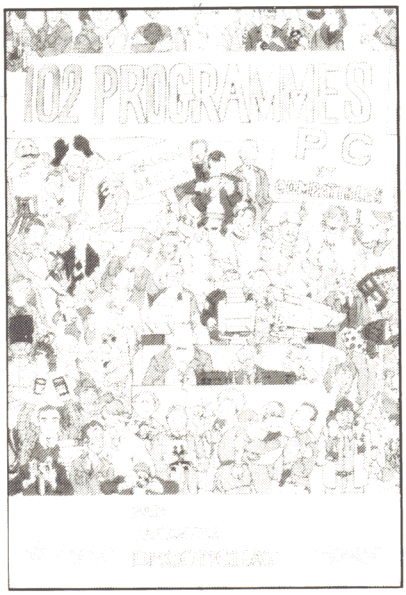
• LE LIVRE DU BASIC SUR PC ET COMPATIBLES (par Jacques BOIGONTIER)

Si vous désirez vous initier rapidement à la pratique du Basic sur PC (ou compatible), n'hésitez pas à accorder votre confiance à Jacques Boigontier. Son livre, bourré d'exemples simples et expliqués, rendra réellement de grands services aux débutants, mais aussi à celles et à ceux qui, connaissant par exemple notre bon vieil Applesoft, ignorent tout du GW-Basic.

Quelques rappels utiles de règles élémentaires vous mettent immédiatement dans le bain, puis les petites routines imaginées par l'auteur deviennent beaucoup plus intéressantes : tracé d'un organigramme, traduction d'un programme en français, éditeur de texte, etc. On étudie sérieusement tous les mots-clés en moins de 200 pages tout en apprenant à tracer des histogrammes et des courbes, à produire des sons et à jouer avec la création de fichiers. Une initiation bien menée par un auteur sérieux.

(145 F TTC) ***PSI*** *— BP 86 — 77402 LAGNY-SUR-MARNE*





• 102 PROGRAMMES PC ET COMPATIBLES (par Jacques DECONCHAT)

Une adaptation intelligente de programmes connus, mais qui intéressent toujours les "micromanes" tentés par les jeux sur ordinateur personnel. Car il s'agit essentiellement de jeux : Nombre mystérieux, Jackpot, Jeu de loto, Marienbad, Othello, Dames (sans dames !), pris aléatoirement parmi la longue liste qui, certainement, saura vous mettre en appétit.

L'auteur a particulièrement soigné la partie explications, à laquelle il consacre au moins une page par programme. La lisibilité des listages est bonne et certaines idées pourront être développées avec profit par les utilisateurs. Si apprendre en distrayant est le but avoué de ce livre, je suis persuadé qu'il y parvient. La difficulté y est en effet graduée (elle va du niveau 1 au niveau 5) et on peut acquérir, en programmant, une connaissance enviable du GW-Basic et du Basic2 sur PC, Amstrad PC et compatibles. Notons que les routines non graphiques peuvent être adaptées sur d'autres machines.

(135 F TTC) ***PSI*** *— BP 86 — 77402 LAGNY-SUR-MARNE*

• LETTRE, IMAGE, ORDINATEUR (par F. HOLTZ-BONNEAU)

L'auteur est chercheur à la Direction de la Recherche de l'Institut national de l'audiovisuel. Elle avait déjà publié un essai sur ***L'Image et l'ordinateur***, essai que je n'ai pas lu. Par contre, j'ai lu avec beaucoup d'intérêt ce passionnant travail sur les créations infographiques.

L'ouvrage de Françoise Holtz-Bonneau est abondamment illustré (en noir et en couleur) et ce n'est pas son moindre mérite. Si l'on nous rappelle que l'ordinateur n'était pas primitivement destiné à œuvrer pour l'art, on nous y montre — et avec talent — qu'il est maintenant appelé à se mettre au service de l'image et à celle de l'imagination créative.

De ***Gutenberg et l'ordinateur*** à ***la créatique***, il y a environ 150 pages, grâce auxquelles le lecteur fait rapidement le tour de la question.

Vous ne vous endormirez certainement pas sur ce livre... sauf, bien sûr, si vous ne vous intéressez absolument pas aux arts typographiques, graphiques, plastiques et audiovisuels.

(195 F TTC)

EDITIONS HERMES *— 51, rue Rennequin — 75017 PARIS*

Chasseur d'Images

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

Chez votre marchand de journaux !