tremplin micro

Programmes pour Apple*

Une calculatrice utilisant la souris

Les tables de formes

Amper command — MIDI pour votre Apple*

M 1631 - 16 - 33,00 F
3791631033006 00160

N° 16 - Bimestriel - Troisième année
5 Septembre - 4 Novembre 1987
254 FB - 11 FS - 33 F

Illustration J.G. Couteau

tremplin micro 16

SOMMAIRE

Avec la collaboration de :

ARGOS, Claude AUBRY, Marcel COTTINI, François GALLET, Roland JOST, Yvan KŒNIG, CLUB MICROTEL MOURENX, NESTOR, Jean-Paul VERPEAUX.

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.

EN TOUTE FRANCHISE

Où en êtes-vous ? Quels sont vos projets immédiats en matière d'informatique personnelle ? Avez-vous l'intention d'écrire des programmes et, si oui, en utilisant quels langages ? Vous êtes-vous laissé tenter par l'Apple *IIGS* ou avez-vous préféré conserver votre Apple *II* ?

J'aurais pu formuler ces demandes — et beaucoup d'autres — sous forme de questionnaire. Peut-être m'y résoudrai-je plus tard. Pour l'instant, je préfère susciter des lettres que je lirai, je vous le promets, avec une grande curiosité. J'espère que vous mettrez aussi ce courrier à profit pour me dire ce que vous pensez du contenu actuel de *TREMPLIN MICRO.*

Bon nombre de nos correspondant(e)s — acquéreurs d'un Apple *IIGS* — se disent "désorientés" par cette machine. D'autres commencent à en utiliser les ressources et écrivent même des routines intéressantes. Nous publierons évidemment les meilleures dans de prochains numéros, mais nous tenons à ne pas décevoir les possesseurs d'Apple *IIe* et *IIc,* et nous continuerons donc à donner la priorité à des programmes fonctionnant sur TOUTES les machines de la gamme APPLE *II.* D'aucuns nous ont reproché d'accorder une place trop importante au langage machine, mais la majorité de nos Lectrices et Lecteurs (du moins celles et ceux qui nous écrivent) approuve cette démarche et *TREMPLIN MICRO* se félicite d'avoir incité des centaines et des centaines d'Applemaniaques à s'intéresser à la programmation du 65Ø2, puis à celle du 65CØ2. Je souhaite personnellement que leur nombre grandisse encore : manipuler les octets constitue un jeu absolument passionnant et il est dommage, quand on a la chance de disposer d'un micro aussi convivial que l'Apple *II,* de se priver de ce plaisir. Bien sûr, ce n'est pas aussi facile que le prétendent les génies de l'informatique (parfaitement, elle en compte quelques-uns !), mais on y parvient avec un peu de persévérance et avec l'aide de petits bouquins comme *"Le 65Ø2 pas à pas", "Clins d'œil au 65Ø2"* et nos fascicules de *"Routines".*

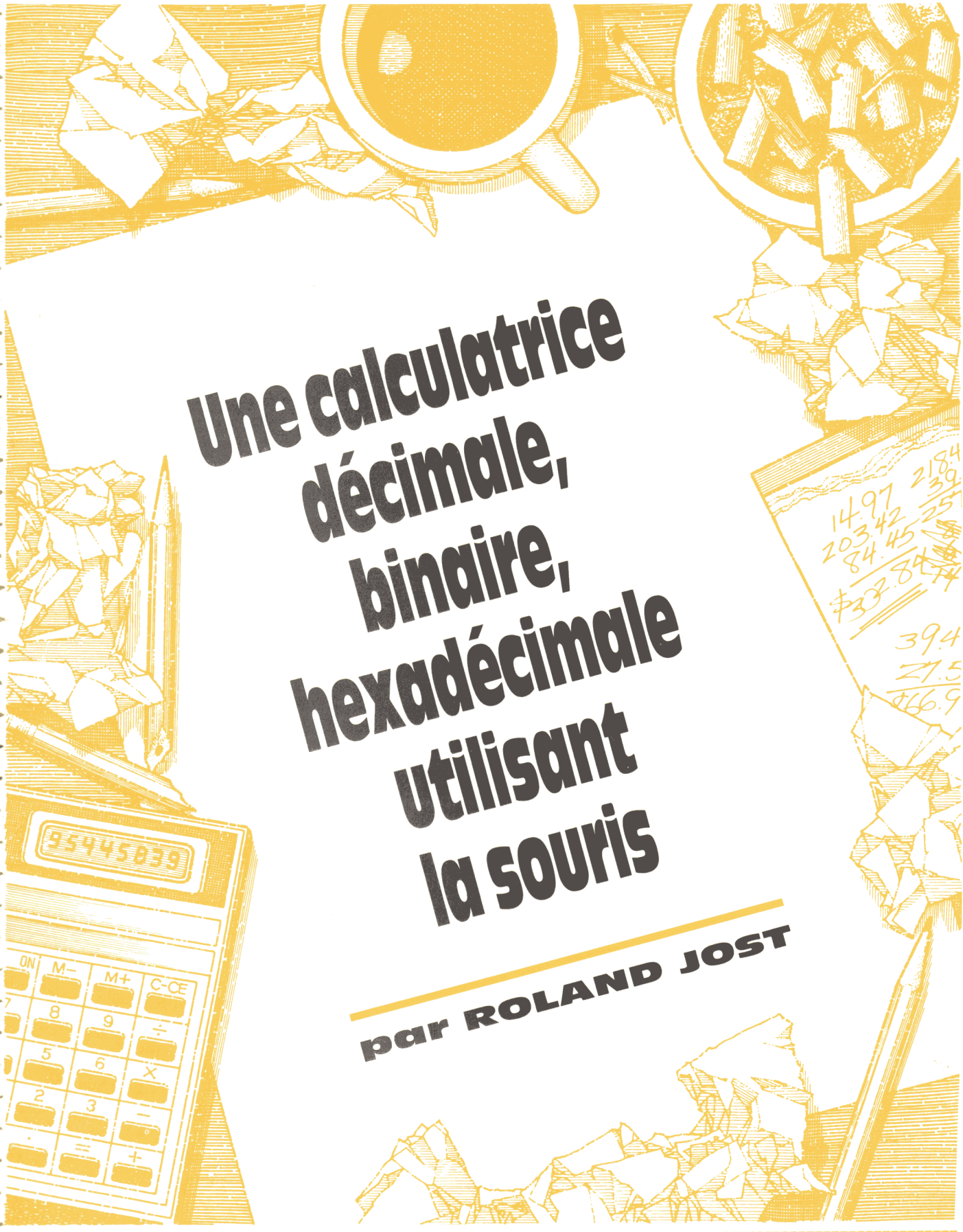
A l'intention des Lectrices et Lecteurs disposant à la fois d'un Apple et d'un IBM PC ou compatible, nous signalons les ouvrages destinés au second, ainsi que certains logiciels très performants et d'un prix abordable (c'est le cas de ceux régulièrement lancés par Borland). Il semble que cette entorse au TOUT-APPLE ne provoque pas trop de réactions épidermiques chez les inconditionnels de la Pomme. On notera que nous parlons peu du Mac, sauf dans les rubriques "livres" : à vous de nous dire ce que vous en pensez.

TREMPLIN MICRO désire rester une revue particulièrement vouée à la programmation, et singulièrement à celle des machines de la longue série des APPLE *II.* L'arrivée du *GS,* ordinateur personnel hybride, modifie certes les données : est-ce un Apple *II* ou un petit Mac ? La question restera encore posée quand, enfin débridé, le *GS PLUS* nous montrera ce qu'il est possible de faire avec un 65C816, les outils du Mac... et des logiciels à la hauteur des ambitions de la technique et de la Rom.

En attendant, programmons le 65CØ2 et considérons le *GS* comme un beau joujou, capable d'émuler le processeur de son prédécesseur... ou de jouer les petits Mac, comme un grand !

BONNE RENTRÉE, À VOS CLAVIERS ET TRAITEMENTS DE TEXTES POUR ÉCRIRE À *"TREMPLIN MICRO"*... ET RENDEZ-VOUS ICI, DANS DEUX MOIS !

Guy-HACHETTE.



Une calculatrice décimale, binaire, hexadécimale utilisant la souris

par ROLAND JOST

Roland JOST

VOTRE CALCULATRICE

La calculatrice JR86 n'est pas une calculatrice comme les autres. Elle réside dans votre APPLE *II* et peut être mise en œuvre grâce à la souris. JR86 utilise la notation polonaise inverse et la pile chères à Hewlett-Packard. En mode décimal, elle donne accès à toutes les fonctions mathématiques usuelles (les fonctions trigonométriques sont réduites au sinus, cosinus et tangente ; les angles sont à exprimer en degrés). En mode hexadécimal et binaire, les calculs se font sur des entiers positifs inférieurs ou égaux à 65535 ($FFFF en base 16 ou %11111111 11111111 en base 2). De plus JR86 permet d'effectuer des opérations sur un octet : opérations logiques AND, ORA et EOR, décalages et rotations ASL, LSR, ROR et ROL.

MISE EN ŒUVRE :

A. Taper le programme BASIC (liste 1) et le sauver sous le titre *JR86.*

B. Nous ne publions pas le source de la routine SP.CALC, beaucoup trop long, mais il figure sur la disquette *Tremplin Micro.* Vous devrez donc taper le code directement, comme suit (liste 2) :
CALL - 151
6000 : 20 BE DE 20 E3 DF 85 19 etc...
Sauver par un **BSAVE SP.CALC.0,A$6000, L$156.**

C. Taper la liste 3 (table de formes) :
CALL - 151
7000 : 18 00 32 00 42 00 etc...
Sauver : **BSAVE TABCALC.0,A$7000,L$1B2.**

D. Entrer enfin la liste 4 (représentation graphique de la calculatrice) :
CALL - 151
6039 : A9 20 85 04 20 63 etc...
Sauver : **BSAVE CALC.COMP.0,A$6039, L$C2F.**
CALC.COMP.0 est long à entrer au clavier, mais c'est indispensable et le résultat en vaut la peine...

E. Vérifier les programmes (ce qui est très facile quand on passe par la disquette SIGNATURE de notre ami Yvan KŒNIG).

UTILISER JR86 :

L'utilisation de la calculatrice en mode décimal ne présente aucune difficulté particulière : entrer un nombre en cliquant les touches numériques, puis cliquer la touche de la fonction à exécuter. Les utilisateurs de calculatrices HP se retrouveront en terrain connu. Pour les autres, rappelons que la notation polonaise inverse permet de se passer de parenthèses et que, par exemple, le calcul de (1 + sin (45)) * 3 s'effectue comme suit :

```
45          (l'angle est exprimé en degrés)
SIN
1
+
3
*
```

Résultat = 5.12131894.

Voici maintenant quelques applications plus spécifiques à la calculatrice JR86 :

En plus des opérations mathématiques classiques en décimal, la JR86 permet de travailler en base 2 ou 16, et aussi d'effectuer des opérations sur un octet. En voici quelques exemples :

1° Conversion d'une adresse négative en nombre positif :

Reliquat des temps héroïques, on trouve encore dans de nombreux listings des commandes du genre PEEK (- 16384) ou CALL - 31ØØ. Pour convertir ces adresses négatives en nombres positifs il suffit d'effectuer les opérations suivantes :

Prenons l'exemple de - 16384, adresse de lecture du clavier
16384
CHS ——» _16384
HEX ——» $CØØØ
DEC ——» 49152

Effectivement un PEEK (- 16384) et un PEEK (49152) sont équivalents en Applesoft.

2° Les opérations logiques : AND, ORA et EOR

JR86 permet d'effectuer des opérations logiques sur des nombres inférieurs à 256, donc codés sur un octet en hexadécimal (8 bits en binaire).

EXEMPLE : faire un ET logique (AND).

```
HEX     pour passer en mode hexa
CØ      (192 en décimal)
ENTER
7F      (127 en décimal)
AND
résultat = 4Ø.
```

3° Les décalages (ASL, LSR) et rotations (ROR, ROL) :

Le microprocesseur 65Ø2 (ou 65CØ2) possède les 4 instructions suivantes :
ASL : décale les bits d'un octet vers la gauche. Le bit 7 est perdu, un zéro rentre dans le bit Ø.
LSR : décale les 8 bits vers la droite : le bit Ø est perdu, un zéro rentre à gauche dans le bit 7.
ROR : rotation à droite d'un octet : la retenue entre dans le bit 7 et le bit Ø tombe dans la retenue.
ROL : rotation à gauche d'un octet : la retenue va dans le bit Ø, le bit 7 sortant tombe dans la retenue.

EXEMPLES :

```
1       (ENTER)
ASL     : 2
ASL     : 4
ASL     : 8
ASL     : 16
etc.
```

En effet ASL effectue une multiplication par 2, LSR une division par 2.

Pour les rotations, le résultat dépend de la valeur de la retenue au départ. Pour mettre la retenue à zéro (donc simuler l'instruction CLC) faire comme suit :
Ø (ENTER)
LSR : le zéro tombe dans la retenue.

Pour mettre la retenue à un (SEC) on fera :
1 (ENTER)
LSR : un 1 tombe dans la retenue.

On pourra maintenant entrer un nombre, puis effectuer l'opération ROL ou ROR.

QUELQUES EXPLICATIONS TECHNIQUES :

- **Le programme JR86 :**
  Le programme BASIC effectue le chargement des différents modules en Assembleur, gère le déplacement de la souris, réalise la saisie des nombres et effectue les différentes opérations mathématiques (sauf les opérations logiques).
- **Notons au passage deux routines :**
  — Ligne 425 : conversion d'un nombre hexa de 4 chiffres, stocké en mémoire à partir de l'adresse A1, en nombre décimal.
  — Ligne 43Ø : conversion de binaire en décimal d'un nombre de 16 bits, stocké en mémoire à partir de l'adresse A2.

LE PROGRAMME SP.CALC.0 :

Ce programme en Assembleur 65Ø2 était nécessaire pour permettre un affichage rapide en mode haute résolution. Il permet de plus la conversion rapide de nombres hexadécimaux ou binaires en décimal (comparer avec la lenteur relative de l'opération inverse effectuée en BASIC par les lignes 425 ou 43Ø) et l'accès aux opérations logiques. Le source de ce programme est bien commenté et le lecteur pourra s'y reporter pour plus de détails. *(Suite page 6)*

TABCALC.0 :

C'est une table de formes renfermant les caractères à afficher en haute résolution (flèche souris, chiffres Ø à 9, lettres A à F, espace, % et $).

CALC.COMP.0 :

L'image graphique de la calculatrice est stockée sur disque sous forme condensée. Cette image est décompressée après chargement par un petit programme assembleur logé entre $6Ø39 et $6Ø8A (technique empruntée à la revue NIBBLE).

Le lecteur qui doit saisir les divers programmes au clavier économise ainsi l'entrée d'environ 4K de nombres hexadécimaux supplémentaires.

LA PROGRAMMATION DE LA SOURIS APPLE :

Même après lecture du manuel de la souris, il n'est pas évident de programmer la souris. Voici les différentes instructions à incorporer dans un programme BASIC :

1° Recherche de la carte souris

Un programme utilisant la souris doit pouvoir détecter la présence de celle-ci. APPLE recommande de mettre la carte dans le port 4, mais ce n'est pas obligatoire, et le programme appelant doit pouvoir localiser le port contenant l'interface. Si l'interface souris est dans le port n l'octet $CnØC contient $2Ø et l'octet $CnFB = $D6. Il suffit donc de tester le contenu de ces 2 octets pour n compris entre 1 et 7. Ceci est réalisé en ligne 875.

2° Activation de la souris

Avant toute utilisation il faut exécuter les instructions :

PRINT CHR$ (4)"PR£"SL : où SL est le numéro de port de l'interface Souris.

PRINT CHR$(1) : pour activer l'interface.

PRINT CHR$(4)"PR£Ø" : pour aiguiller les sorties vers l'écran.

3° Lecture de la souris

PRINT CHR$(4)"IN£"SL : input provenant de la souris

INPUT""; H,V,S : : saisie de la position de la souris

H = position horizontale : $Ø = < H <= 1Ø23$

V = position verticale : $Ø = < V <= 1Ø23$

S = statut du bouton : $-4 <= S <= 4$

4° Désactivation de la souris

PRINT CHR$(4)"IN£Ø" : pour réactiver le clavier

PRINT CHR$(4)"PR£"SL : envoi de caractères vers la carte souris

PRINT CHR$(Ø) : désactive la souris

PRINT CHR$(4)"PR£"Ø : envoi des caractères vers l'écran.

ET POUR CONCLURE VOICI QUELQUES MISES EN GARDE :

A. Les touches A,B,C,D,E et F ne correspondent pas à des registres, mais aux chiffres hexadécimaux 1Ø à 15. Elles ne sont à utiliser qu'en mode hexadécimal.

B. Les décalages et rotations ne peuvent être effectués que sur des entiers inférieurs ou égaux à 255. Sinon affichage du message d'erreur (EEEEE clignotant).

C. Les autres opérations en base 16 ou 2 ne portent que sur des entiers pouvant être codés sur 2 octets, c'est-à-dire strictement inférieurs à 65536.

Si vous ne disposez pas de la souris APPLE, utilisez les flèches pour déplacer le curseur et la touche (ENTER) pour valider une commande. Bien sûr, avec la souris, C'EST MIEUX !

LISTAGE 1 — JR86

```
100 REM ***** JR86 : CALCULATRIC
    E SOURIS *****
105 REM *****   Roland JOST  (C)
     1986     *****
110 GOSUB 715: GOTO 275                C0D0
115 REM ***** SOUS-PROGRAMMES **
    ***
120 REM ***** AFFICHAGE DES NOMB
    RES *****
125 GOSUB 185: GOSUB 155               BDD3
130 IF BASE = 16 THEN  CALL AH,H
    $: CALL HEX,X: CALL PR,H$,11
    6,8,1: RETURN                      070F
135 IF BASE = 2 THEN  CALL AB,B$
    : CALL BIN,X: CALL PR,B$,44,
    8,1: RETURN                        DE8C
140 A$ =  STR$ (X)                     93C2
145 XH = 140 - 6 *  LEN (A$): CA
    LL PR,A$,XH,8,1                    6FED
150 RETURN                             63B1
155 REM ***** AFFICHAGE DU MODE
    *****
160 HCOLOR= 0: DRAW 15 AT 38,8:
    HCOLOR= 3                          8F89
165 IF BASE = 16 THEN  DRAW 24 A
    T 38,8: RETURN                     FB3C
170 IF BASE = 2 THEN  DRAW 23 AT
    38,8: RETURN                       D606
175 RETURN                             63B1
180 REM ***** EFFACEMENT *****
185 HCOLOR= 0: CALL PR,V$,44,8,1
    : HCOLOR= 3: RETURN                100F
190 REM ***** SAISIE D'UN NOMBRE
     *****
195 IF BASE <  > 2 THEN 215            36F6
200 IF Q < 2 THEN XX = XX * 2 +
    Q                                  F60A
205 GOSUB 185: GOSUB 155: CALL A
    B,B$: CALL BIN,XX: CALL PR,B
    $,44,8,1                           E378
210 RETURN                             63B1
215 IF BASE <  > 10 THEN 245           EB28
220 IF Q > 9 THEN 235                  A664
225 XX =  VAL (A$)                     931B
230 GOSUB 185: GOSUB 155               BDD3
235 CALL PR,A$,140 - 6 *  LEN (A
    $),8,1                             CFA3
240 RETURN                             63B1
245 XX = XX * 16 + Q                   747A
250 GOSUB 185: GOSUB 155: CALL A
    H,H$: CALL HEX,XX: CALL PR,H
    $,116,8,1                          96C6
255 RETURN                             63B1
260 REM ***** TRAITEMENT ERREURS
     *****
265 FOR T = 1 TO 5: PRINT  CHR$
    (7);: GOSUB 185: CALL PR,E$,
    70,8,1: NEXT : GOSUB 465: GO
    TO 275                             CA0B
270 REM ***** DEPLACEMENTS CURSE
    UR *****
275 IF MS = 0 THEN 290                 5CAC
280 PRINT D$"IN£"SL: REM ACTIVE
    LECTURE SOURIS                     59BF
285 REM LECTURE
290 H1 = H:V1 = V                      A278
295 IF MS = 0 THEN  GET R$:R =
    ASC (R$):H = H - 28 * (R = 8
    ) + 28 * ((R = 21) OR (R = 3
    2)):V = V + 13 * (R = 10) -
    13 * (R = 11): GOTO 310            BF1E
300 INPUT "";H,V,S                     B54C
305 H = 28 *  INT (H / 25.575) -
    3:V = 13 *  INT (V / 32)           6EED
310 IF H < 53 THEN H = 53              B272
315 IF H > 137 THEN H = 137            89D6
320 IF V < 26 THEN V = 26              AC8E
325 IF V > 182 THEN V = 182            02F2
330 IF  ABS (S) = 2 OR R = 13 TH
    EN GOSUB 350                       5F87
335 IF H = H1 AND V = V1 THEN 27
    5                                  B11A
340 XDRAW 16 AT H1,V1: XDRAW 16
    AT H,V                             BAB2
345 GOTO 275                           3F49
350 REM ON A CLIQUE
355 PRINT  CHR$ (7);                   7364
360 T1 = TCHE:TCHE = 4 * (V / 13
    - 2) + (H - 53) / 28: IF TCH
    E = 51 THEN 925                    E8E2
365 IF TCHE < 28 OR TCHE = 32 OR
    TCHE = 36 OR TCHE = 40 OR TC
    HE = 44 OR TCHE = 48 THEN  G
    OSUB 415: GOTO 375                 946E
370 IF T1 < 28 OR T1 = 32 OR T1
    = 36 OR T1 = 40 OR T1 = 44 O
    R T1 = 48 THEN  GOSUB 395:T
    = Z:Z = Y:Y = X:XX = 0             1151
375 IF TCHE > 29 THEN 385              9B6F
380 ON TCHE + 1 GOTO 450,455,460
    ,465,470,475,480,485,490,495
    ,500,505,510,515,520,525,530
    ,535,540,545,550,555,560,565
    ,570,575,580,585,590,595           FFCF
```

(Suite page 8)

LISTAGE 1 JR86

```
385 ON TCHE - 29 GOTO 600,605,61
    0,615,620,625,630,635,640,64
    5,650,655,660,665,670,675,68
    0,685,690,700,705                35FA
390 REM
395 REM ***** MISE A BLANC DES C
    HAINES *****
400 IF BASE = 10 THEN A$ = "": R
    ETURN                            1621
405 IF BASE = 16 THEN  FOR I = 0
    TO 3: POKE A1 + I,48: NEXT :
    RETURN                           D736
410 IF BASE = 2 THEN  FOR I = 0
    TO 15: POKE A2 + I,48: NEXT
    : RETURN                         C735
415 REM ***** EVALUATION DE X **
    ***
420 IF BASE = 10 THEN X =  VAL (
    A$): RETURN                      036B
425 IF BASE = 16 THEN X = 0: FOR
    L = 0 TO 3:Z =  PEEK (A1 + L
    ):Z = Z - 48 * (Z > 47 AND Z
    < 58) - 55 * (Z > 64 AND Z <
    71):ZZ(L) = Z: NEXT : FOR I
    = 0 TO 3:X = X + ZZ(I) * (16
    ^ (3 - I)): NEXT : RETURN        8B3F
430 IF BASE = 2 THEN X = 0: FOR
    L = 0 TO 15:Z =  PEEK (A2 +
    L):Z = Z - 48 * (Z > 47 AND
    Z < 50):ZZ(L) = Z: NEXT : FO
    R I = 0 TO 15:X = X + ZZ(I)
    * (2 ^ (15 - I)): NEXT : RET
    URN                              0A26
435 REM ADDITION A LA PILE
440 T = Z:Z = Y:Y = X: RETURN        4EE1
445 REM ***** OPERATIONS *****
450 BASE = 16: GOTO 120              BACA
455 BASE = 10: GOTO 120              0CC4
460 BASE = 2: GOTO 120               4995
465 X = 0: GOSUB 395: GOSUB 185:
    RETURN                           8B56
470 X =  INT (X): GOTO 120           E71C
475 X =  ABS (X): GOTO 120           D01D
480 X =  SQR (X): GOTO 120           6E23
485 X = X * X: GOTO 120              C71A
490 X =  LOG (X) /  LOG (10): GO
    TO 120                           957E
495 X =  LOG (X): GOTO 120           9C25
500 X =  EXP (X): GOTO 120           2926
505 X = Y ^ X: GOTO 120              F81D
510 X =  SIN (X * PI / 180): GOT
    O 120                            BAEF
515 X =  COS (X * PI / 180): GOT
    O 120                            C9EE
520 X =  TAN (X * PI / 180): GOT
    O 120                            81F0
525 X =  ATN (X * PI / 180): GOT
    O 120                            29F1
530 T = Z:Z = Y:Y = X:X = PI: GO
    TO 120                           AA69
535 X =  - X: GOTO 120               30C1
540 WW = X:X = Y:Y = Z:Z = T:T =
    WW: GOTO 120                     298A
545 WW = T:T = Z:Z = Y:Y = X:X =
    WW: GOTO 120                     048A
550 POKE 6,X: CALL LS:X =  PEEK
    (6): GOTO 120                    6F1B
555 X = 1 / X: GOTO 120              F8F4
560 WW = X:X = Y:Y = WW: GOTO 12
    0                                3E1A
565 : RETURN                         63EB
570 POKE 6,X: CALL AS:X =  PEEK
    (6): GOTO 120                    CC10
575 POKE 6,X: POKE 7,Y: CALL AN:
    X =  PEEK (6): GOTO 120          9CBA
580 POKE 6,X: POKE 7,Y: CALL EO:
    X =  PEEK (6): GOTO 120          F3BF
585 POKE 6,X: POKE 7,Y: CALL RA:
    X =  PEEK (6): GOTO 120          5FBE
590 POKE 6,X: CALL RL:X =  PEEK
    (6): GOTO 120                    4C1A
595 Q = 13: GOTO 190                 7504
600 Q = 14: GOTO 190                 4E05
605 Q = 15: GOTO 190                 4B06
610 POKE 6,X: CALL RR:X =  PEEK
    (6): GOTO 120                    4320
615 Q = 10: GOTO 190                 5901
620 Q = 11: GOTO 190                 9102
625 Q = 12: GOTO 190                 9B03
630 X = Y - X:Y = Z:Z = T: GOTO
    120                              EC8F
635 A$ = A$ + "7":Q = 7: GOTO 19
    0                                74EE
640 A$ = A$ + "8":Q = 8: GOTO 19
    0                                77F0
645 A$ = A$ + "9":Q = 9: GOTO 19
    0                                EFF2
650 X = Y + X:Y = Z:Z = T: GOTO
    120                              328E
655 A$ = A$ + "4":Q = 4: GOTO 19
    0                                68E8
660 A$ = A$ + "5":Q = 5: GOTO 19
    0                                15EA
665 A$ = A$ + "6":Q = 6: GOTO 19
    0                                28EC
670 X = Y * X:Y = Z:Z = T: GOTO
```

```
    120                                   E790
675 A$ = A$ + "1":Q = 1: GOTO 19
    0                                     68E2
680 A$ = A$ + "2":Q = 2: GOTO 19
    0                                     53E4
685 A$ = A$ + "3":Q = 3: GOTO 19
    0                                     D7E6
690 X = Y / X: GOTO 120                   C71C
695 X = Y / X:Y = Z:Z = T: GOTO
    120                                   3091
700 A$ = A$ + "0":Q = 0: GOTO 19
    0                                     E8E0
705 A$ = A$ + ".": GOTO 190               6D53
710 REM ***** INITIALISATION ***
    **
715 DIM ZZ(16)                            31F2
720 D$ =  CHR$ (4)                        B3A4
725 E$ = "EEEEEE"                         A81B
730 H$ = "1215"                           0549
735 B$ = "0001000111001101"               5981
740 V$ = " ----------------"              7F8E
745 BASE = 10                             034C
750 AH = 24576: REM recherche ad
    resse de H$                           E961
755 AB = 24603: REM recherche ad
    resse de B$                           E852
760 AN = 24630: REM AND                   B45E
765 EO = 24637: REM EOR                   E86A
770 RA = 24644: REM ORA                   BE67
775 AS = 24651: REM ASL                   A866
780 LS = 24654: REM LSR                   4174
785 RL = 24673: REM ROL                   B774
790 RR = 24681: REM ROR                   7379
795 BIN = 24689: REM conversion
    décimal-binaire.                      30B6
800 HEX = 24737: REM conversion
    décimal-hexadécimal                   CEBC
805 PR = 24771: REM écriture HGR
    .                                     4377
810 PI = 3.14159                          B2CE
815 SCALE= 1: ROT= 0: HCOLOR= 3           B2CB
820 ONERR  GOTO 260                       75E8
825 GOSUB 865: GOSUB 900: HGR :
    POKE - 16302,0: PRINT D$"BRU
    N CALC.COMP.0"                        297A
830 PRINT D$"BLOADTABCALC.0,A$70
    00": POKE 232,0: POKE 233,11
    2                                     8999
835 PRINT D$"BLOAD SP.CALC.0"             AA2A
840 CALL AH,H$:A1 =  PEEK (250)
    + 256 *  PEEK (251)                   93ED
845 CALL AB,B$:A2 =  PEEK (235)
    + 256 *  PEEK (236)                   EDE8
850 GOSUB 185                             FC4E
855 S = 4                                 E257
860 H = 53:V = 26: XDRAW 16 AT H
    ,V:H1 = H:V1 = V                      EFBF
865 REM ***** RECHERCHE DE LA CA
    RTE SOURIS
870 REM :$CN0C = 32 ET $CNFB = 2
    14 ( SOFT SOURIS-
875 FOR I = 1 TO 7: IF  PEEK (49
    164 + I * 256) = 32 AND  PEE
    K (49403 + I * 256) THEN SL
    = I:I = 7: NEXT : GOTO 890            6A56
880 NEXT I                                9DCB
885 PRINT "PAS DE SOURIS . UTILI
    SER LES TOUCHES FLECHES DU C
    LAVIER ":MS = 0: RETURN               4E8C
890 PRINT "INTERFACE SOURIS EN S
    LOT "SL:MS = 1                        6B43
895 RETURN                                63B1
900 REM ***** ACTIVATION SOURIS
    *****
905 IF MS = 0 THEN  RETURN                78C2
910 PRINT D$"PR£"SL: PRINT  CHR$
    (1): REM ACTIVE LA SOURIS             4027
915 PRINT D$"PR£0": REM AFFICHAG
    E ECRAN                               3A5B
920 RETURN                                63B1
925 REM ***** DESACTIVATION SOUR
    IS ET FIN *****
930 PRINT                                 75BA
935 PRINT D$"IN£0": REM ACCEPTE
    INPUT DU CLAVIER                      C050
940 PRINT D$"PR£"SL: PRINT  CHR$
    (0): REM DESACTIVE LA SOURIS          4626
945 PRINT D$"PR£0": REM ECRAN             3A5B
950 TEXT                                  1389
955 END                                   0180
```

SIGNATURE

Cette disquette *TREMPLIN MICRO* permet une vérification facile des programmes (BASIC et LM) insérés dans la revue. Bon nombre de nos lecteurs l'utilisent déjà et s'en félicitent.

Utiliser le bulletin de commande, à la fin de la revue

SP.CALC.Ø,A$6ØØØ,L$Ø156

LISTE 2

```
6000: 20 BE DE 20 E3 DF 85 19 84 1A A0 00 B1 19 85 FC C8 B1 19 85 FA C8 B1 19   5368
6018: 85 FB 60 20 BE DE 20 E3 DF 85 19 84 1A A0 00 B1 19 85 ED C8 B1 19 85 EB   0998
6030: C8 B1 19 85 EC 60 A5 06 25 07 85 06 60 A5 06 45 07 85 06 60 A5 06 05 07   F7BE
6048: 85 06 60 06 06 60 46 06 20 54 60 60 90 06 A9 01 85 09 B0 04 A9 00 85 09   CE90
6060: 60 46 09 26 06 20 54 60 60 46 09 66 06 20 54 60 60 20 BE DE 20 67 DD 20   A73E
6078: 52 E7 A0 07 66 51 90 04 A9 31 B0 02 A9 30 91 EB 88 C0 FF D0 EF A0 0F 66   5D27
6090: 50 90 04 A9 31 B0 02 A9 30 91 EB 88 C0 07 D0 EF 60 20 58 FC 20 BE DE 20   8683
60A8: 67 DD 20 52 E7 A5 51 A6 50 20 41 F9 A0 03 B9 00 04 29 7F 91 FA 88 C0 FF   1EBD
60C0: D0 F4 60 20 BE DE 20 E3 DF 85 19 84 1A A0 00 84 18 84 FC B1 19 85 FB C8   59CC
60D8: B1 19 48 C8 B1 19 85 FE 68 85 FD 20 BE DE 20 67 DD 20 52 E7 A5 50 85 06   070A
60F0: A5 51 85 07 20 F5 E6 86 FC 20 F5 E6 86 E7 A9 06 85 EC A4 18 B1 FD C9 20   89DA
6108: D0 02 A9 3F C9 2E D0 02 A9 3C C9 45 D0 02 A9 3B C9 2B D0 02 A9 3D C9 2D   12CE
6120: D0 02 A9 3E C9 30 D0 02 A9 3A 38 E9 30 AA 20 30 F7 A5 FC A6 06 A4 07 20   02C1
6138: 11 F4 A5 F9 20 05 F6 A4 06 18 98 65 EC 85 06 C4 06 90 02 E6 07 E6 18 A6   8DE1
6150: FB E4 18 D0 AD 60                                                         F1D4
```

TABCALC.Ø,A$7ØØØ,L$Ø1B2

LISTE 3

```
7000: 18 00 32 00 42 00 53 00 64 00 74 00 85 00 96 00 A7 00 B8 00 C9 00 D9 00   CCD3
7018: EA 00 F1 00 FE 00 09 01 1E 01 2F 01 3F 01 50 01 60 01 70 01 81 01 91 01   73A9
7030: A2 01 09 6D DA DF 4A 69 DA DF 4A 69 1A 3F 9F 00 00 00 29 2D D5 DF 53 49   B68F
7048: D5 3F BF 69 89 3B 3F 17 00 00 00 29 2D D5 DF 53 49 D5 3F 9F 49 A9 3B 3F   921B
7060: 17 00 00 00 69 89 FB BB 69 A9 3B 3F 57 49 D5 DF 13 00 00 00 29 2D D5 DB   24B8
7078: 57 2D AD FB 9B 49 A9 3B 3F 17 00 00 00 29 2D D5 DB 57 2D AD FB BB 69 A9   0149
7090: 3B 3F 17 00 00 00 29 2D D5 DF 53 09 8D 1B DF 4A 4D DA FB 02 00 00 00 29   0415
70A8: 2D D5 DF 57 2D AD FB BB 69 A9 3B 3F 17 00 00 00 29 2D D5 DF 57 2D AD FB   75A1
70C0: 9B 49 A9 3B 3F 17 00 00 00 29 2D D5 DF 57 4D D5 DF 57 4D D5 3F BF 00 00   18F7
70D8: 00 29 2D D5 DB 57 2D 8D DB BB 69 89 3B 3F 17 00 00 00 92 52 09 8D 00 00   42AA
70F0: 00 52 09 8D 3B FF 4A 69 DA DB 02 00 00 00 92 09 2D D5 DB 53 49 11 00 00   30B1
7108: 00 2D 2D 35 3F 3F 37 2D 2D 35 3F 3F 37 2D 2D 35 3F 3F 37 00 00 00 2D 6D   F566
7120: DA 3B 37 2D 4D DA DF 4E 09 15 DF 9B 00 00 00 09 6D 1A DF 57 2D AD FB BB   F3BB
7138: 69 A9 FB BB 00 00 00 29 6D 1A DF 57 2D 8D FB BB 69 A9 1B 3F 17 00 00 00   7BA1
7150: 09 6D 1A DF 57 4D D1 DB 57 4D D5 3B 9F 00 00 00 29 6D 1A DF 57 4D D5 DF   DFF9
7168: 57 4D D5 3B BF 00 00 00 29 2D D5 DB 57 2D 8D DB BB 69 89 3B 3F 17 00 00   C9A3
7180: 00 29 2D D5 DB 57 2D 8D DB BB 69 89 DB BB 00 00 00 29 4D 1E DF 57 09 8D   F195
7198: DB 9F 69 A9 3B DF 06 00 00 00 69 A9 FB BB 29 2D D5 DF 57 4D D5 DF 17 00   FFED
71B0: 00 00                                                                     0000
```

CALC.COMP.Ø,A$6039,L$ØC2F

LISTE 4

```
6039: A9 20 85 04 20 63 60                                                      4735
6040: B1 02 F0 0B 91 00 20 84 60 20 76 60 D0 F2 60 20 84 60 B1 02 AA 98 91 00   0CE5
6058: 20 76 60 CA D0 F7 20 84 60 D0 DD A5 04 85 01 A9 00 85 00 A9 8B 85 02 A9   2BF9
6070: 60 85 03 A0 00 60 E6 00 D0 02 E6 01 A5 01 38 E9 20 C5 04 60 E6 02 D0 C6   D115
6088: E6 03 60 00 04 7E 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 00 17   51D2
60A0: 01 5F 37 7A 01 5F AB 7F 01 3F 2B 7A 01 6F 36 7A 01 01 00 16 01 0F 23 7C   7367
60B8: 01 7F 43 7F 01 7F 63 7F 01 7F 43 7F 01 01 00 1E 41 00 03 60 01 7A 5E 0F   4C92
60D0: 00 01 18 6F 7B 00 03 02 01 00 16 01 00 10 01 00 16 01 00 10 01 00 1E 01   DF78
60E8: 7E 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 00 01 01 00 16 01 7F 3B 7F   75C2
6100: 01 1F AA 7C 01 3F 3B 7F 01 7F 76 7E 01 01 00 16 01 7F FF 7F 01 7F F7 7F   C3C0
6118: 01 7F EB 7F 01 7F EF 7F 01 01 00 1E 01 6F A2 7A 01 0F A3 7C 01 0F 63 76   2D9C
6130: 01 1F 3A 7D 01 01 00 16 01 7E 7F 7F 01 7F FF 7F 01 7F FF 7F 01 7F 7F 7F   EEE6
6148: 01 01 00 16 01 FE FF FF 00 01 FE FF FF 00 01 FE FF FF 00 01 FE FF FF 00   E40C
6160: 01 01 00 1E 01 00 10 01 00 16 01 00 10 01 00 16 01 7E 7F 7F 00 01 7E 7F   B8EB
6178: 7F 00 01 7E 7F 7F 00 01 7E 7F 7F 00 01 01 00 1E 01 5F E5 7E 01 5F DA 7B   3711
6190: 01 7F A2 7D 01 2F F7 7E 01 01 00 16 01 1F 22 7F 01 1F 42 7A 01 1F 2B 7C   5BC0
61A8: 01 5F 62 78 01 01 00 16 01 7F EB 7F 01 7F 6F 7F 01 7F EF 7F 01 7F 5F 7F   C6F6
```

```
61C0: 01 01 00 1E 01 FE FF FF 00 01 FE FF FF 00 01 FE FF FF 00 01 FE FF FF 00   6014
61D8: 01 01 00 16 01 00 10 01 00 16 01 00 10 01 00 1E 01 7E 7F 7F 00 01 7E 7F   82EB
61F0: 7F 00 01 7E 7F 7F 00 01 7E 7F 7F 00 01 01 00 16 01 7F 7F 7F 01 7F 7F 7F   7A8D
6208: 01 7F 7F 7F 01 7F 7F 7F 01 01 00 16 01 7F FB 7F 01 FF E3 7F 01 7F FF 7F   A86E
6220: 01 1D 77 04 01 01 00 1E 03 00 10 01 00 16 01 1F 36 78 01 5F AB 7C 01 3F   B878
6238: 23 78 01 0F 36 78 01 01 00 16 01 6F 2A 7B 01 7F 5B 7F 01 7F 5B 7F 01 7F   15BA
6250: 7B 7F 01 01 00 1E 41 00 03 20 02 4A 02 02 00 01 10 29 0A 00 03 02 01 00   AB18
6268: 16 01 00 10 01 00 16 01 00 10 01 00 1E 01 00 10 01 00 16 01 7F 7F 7F 01   0815
6280: 7F 7F 7F 01 7F 7F 7F 01 7F 7F 7F 01 01 00 16 01 7F FF 7F 01 7F F7 7F 01   7486
6298: 7F E3 7F 01 7F E3 7F 01 01 00 1E 01 6F FA 7A 01 6F BA 7F 01 6F 76 74 01   3ECB
62B0: 5F 3B 7D 01 01 00 16 01 5E 23 7C 01 1F EF 7A 01 7F DF 7F 01 23 45 48 01   B646
62C8: 01 00 16 01 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01 01 00 1E 01   A630
62E0: 00 10 01 00 16 01 00 10 01 00 16 01 00 10 01 00 1E 01 0F ED 7E 01 5F E2   353C
62F8: 78 01 1F A2 7B 01 2F F7 78 01 01 00 16 01 5F 2E 7F 01 5F 4A 7A 01 5F 2B   8D28
6310: 7D 01 5F 6A 7A 01 01 00 16 01 7F F7 7F 01 7F 6F 7F 01 7F E3 7F 01 7F 4F   FEEE
6328: 7F 01 01 00 1E 01 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01 01 00   C999
6340: 16 01 FE FF FF 00 01 FE FF FF 00 01 FE FF FF 00 01 FE FF FF 00 01 01 00   D80C
6358: 16 01 00 10 01 00 1E 01 00 10 01 00 16 01 7E 7F 7F 00 01 7E 7F 7F 00 01   1669
6370: 7E 7F 7F 00 01 7E 7F 7F 00 01 01 00 16 01 7F 7F 7F 01 7E 7F 7F 01 7F 7F   3C0B
6388: 7F 01 21 00 02 01 01 00 1E 01 00 10 01 00 16 01 7F 36 79 01 5F EB 7D 01   61E3
63A0: 3F 2B 79 01 6F 36 79 01 01 00 16 01 0F 2B 7C 01 7F 43 7F 01 7F 63 7F 01   4C76
63B8: 7F 7B 7F 01 01 00 1E 41 00 03 60 01 4A 1E 02 00 01 10 6F 7B 00 03 02 01   79A9
63D0: 00 16 01 FE FF FF 00 01 FE FF FF 00 01 FE FF FF 00 01 FE FF FF 00 01 01   9D0C
63E8: 00 16 01 00 10 01 00 1E 01 00 10 01 00 16 01 7E 7F 7F 00 01 7E 7F 7F 00   5D68
6400: 01 7E 7F 7F 00 01 7E 7F 7F 00 01 01 00 16 01 7F 7F 7F 01 7F 7F 7F 01 7F   BB8E
6418: 7F 7F 01 7F 7F 7F 01 01 00 1E 01 0F F2 7D 01 6F B2 7F 01 0F 77 70 01 5F   E213
6430: 7B 7E 01 01 00 16 01 5E 3B 7B 01 3F EF 7A 01 6B BF 6B 01 3B 6C 2E 01 01   893D
6448: 00 16 01 7F FF 7F 01 7F FB 7F 01 7F E3 7F 01 7F E3 7F 01 01 00 1E 01 00   EAF3
6460: 10 01 00 16 01 00 10 01 00 16 01 00 10 01 00 1E 01 7F 7F 7F 01 7F 7F 7F   9B7B
6478: 01 7F 7F 7F 01 7F 7F 7F 01 01 00 16 01 5F 22 7C 01 5F 5A 7C 01 1F 22 7B   8405
6490: 01 1F 5A 7A 01 01 00 16 01 7F F7 7F 01 7F 6F 7F 01 7F FB 7F 01 7F 5F 7F   CCC8
64A8: 01 01 00 1E 01 5F A3 78 01 7F DE 7D 01 1F AA 7C 01 7F D7 7F 01 01 00 16   B0AA
64C0: 01 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01 01 00 16 01 7E 7F 7F   2A8D
64D8: 00 01 FE FF FF 00 01 FE FF FF 00 01 7E 7F 7F 00 01 01 00 1E 01 00 10 01   D6A9
64F0: 00 16 01 00 10 01 00 16 01 7E 7F 7F 00 01 7E 7F 7F 00 01 7E 7F 7F 00 01   E9B6
6508: 7E 7F 7F 00 01 01 00 1E 01 00 10 01 00 16 01 1F 22 7B 01 1F A2 7C 01 3F   DCFF
6520: 2B 7B 01 6F 36 7B 01 01 00 16 01 2F 2B 7D 01 7F 5B 7F 01 7F 5B 7F 01 7F   A6EB
6538: 7B 7F 01 01 00 1E 41 00 03 60 20 4A 10 02 3C 10 28 4A 00 03 02 01 00 16   FE14
6550: 01 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01 01 00 16 01 FE FF FF   270D
6568: 00 01 FE FF FF 00 01 FE FF FF 00 01 FE FF FF 00 01 01 00 1E 01 00 10 01   C129
6580: 00 16 01 00 10 01 00 16 01 7E 7F 7F 00 01 7E 7F 7F 00 01 7E 7F 7F 00 01   E9B6
6598: 7E 7F 7F 00 01 01 00 1E 01 6F FA 7A 01 6F BA 7F 01 6F 76 72 01 5F 7B 7E   09DA
65B0: 01 01 00 16 01 5E 23 7C 01 3F F7 7D 01 77 00 01 6A 01 33 6C 4C 01 01 00   D89B
65C8: 16 01 7F F7 7F 01 7F EB 7F 01 7F FB 7F 01 7F FB 7F 01 01 00 1E 01 FE FF   1708
65E0: FF 00 01 FE FF FF 00 01 FE FF FF 00 01 FE FF FF 00 01 01 00 16 01 00 10   4D1F
65F8: 01 00 16 01 00 10 01 00 1E 01 7E 7F 7F 00 01 7E 7F 7F 00 01 7E 7F 7F 00   A8BE
6610: 01 7E 7F 7F 00 01 01 00 16 01 7F 7F 7F 01 7F 7F 7F 01 7F 7F 7F 01 7F 7F   A80E
6628: 7F 01 01 00 16 01 7F EB 7F 01 7F 47 7F 01 7F E3 7F 01 7F 47 7F 01 01 00   03F1
6640: 1E 01 5F AB 7F 01 7F 9E 7D 01 5F AB 7B 01 3F ED 7F 01 01 00 16 01 0F 23   1DC0
6658: 7F 01 7F 63 7F 01 7F 43 7F 01 7F 43 7F 01 01 00 16 01 7F FF 7F 01 FF FF   347A
6670: 7F 01 7F FF 7F 01 01 00 02 01 01 00 1E 01 00 10 01 00 16 01 00 10 01 00   B1DB
6688: 16 01 00 10 01 00 1E 01 00 10 01 00 16 01 7F 7F 7F 01 7F 7F 7F 01 7F 7F   9669
66A0: 7F 01 7F 7F 7F 01 01 00 16 01 6F 22 7B 01 7F 5B 7F 01 7F 63 7F 01 7F 43   27A1
66B8: 7F 01 01 00 1E 41 00 03 20 21 4A 10 02 00 01 10 28 4A 00 03 02 01 00 16   771F
66D0: 01 7F 83 7F 01 1F AA 7C 01 3F 3C 7F 01 7F 78 7E 01 01 00 16 01 7F FF 7F   DB4F
66E8: 01 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01 01 00 1E 01 00 10 01 00 16 01 00   6643
6700: 10 01 00 16 01 00 10 01 00 1E 01 6F A2 7A 01 0F A3 7C 01 0F 63 76 01 1F   801B
```

```
6718: 22 7D 01 01 00 16 01 5E 2F 7D 01 3F FB 7A 01 77 FD 63 01 3B 6C 2E 01 01   FE27
6730: 00 16 01 7F E3 7F 01 7F E3 7F 01 7F E3 7F 01 7F E3 7F 01 01 00 1E 01 7F   453E
6748: FF 7F 01 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01 01 00 16 01 FE FF FF 00 01   C28E
6760: FE FF FF 00 01 FE FF FF 00 01 FE FF FF 00 01 01 00 16 01 00 10 01 00 1E   363E
6778: 01 00 10 01 00 16 01 7C 7F 7F 00 01 7E 7F 7F 00 01 7E 7F 7F 00 01 7E 7F   119B
6790: 7F 00 01 01 00 16 01 7F 7F 7F 01 7F 7F 7F 01 7F 7F 7F 01 7F 7F 7F 01 01   D191
67A8: 00 1E 01 5F AB 79 01 7F 9E 7C 01 1F B7 7C 01 3F ED 7F 01 01 00 16 01 6F   F4C3
67C0: 2A 7F 01 7F 5B 7F 01 7F 7B 7F 01 7F 7B 7F 01 01 00 16 01 7F EF 7F 01 FF   9DFD
67D8: E3 7F 01 7F FF 7F 01 1D 75 0E 01 01 00 1E 01 00 10 01 00 16 01 00 10 01   E55B
67F0: 00 16 01 00 10 01 00 1E 01 00 10 01 00 16 01 7E 7F 7F 00 01 7E 7F 7F 00   5D68
6808: 01 7E 7F 7F 00 01 7E 7F 7F 00 01 01 00 16 01 7F 7F 7F 01 7F 7F 7F 01 7F   BB8E
6820: 7F 7F 01 7F 7F 7F 01 01 00 1E 41 00 03 20 4A 79 1E 02 00 01 38 6F 7B 00   9906
6838: 03 02 01 00 16 01 7F D5 7F 01 5F AB 7F 01 3F 3B 7F 01 7F 36 7C 01 01 00   57A8
6850: 16 01 7F FF 7F 01 7F E3 7F 01 7F E3 7F 01 7F E3 7F 01 01 00 1E 01 00 10   9EEB
6868: 01 00 16 01 00 10 01 00 16 01 00 10 01 00 1E 01 7F 7F 7F 01 7F 7F 7F 01   616C
6880: 7F 7F 7F 01 7F 7F 7F 01 01 00 16 01 1E 22 7B 01 1F FA 7A 01 6B FB 6F 01   BC3A
6898: 23 6D 28 01 01 00 16 01 7F F7 7F 01 7F EF 7F 01 7F EF 7F 01 7F EB 7F 01   4F8D
68B0: 01 00 1E 01 0F AD 7C 01 1F E2 78 01 1F A2 7C 01 2F F7 78 01 01 00 16 01   4DC8
68C8: 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01 01 00 16 01 FE FF FF 00   C00C
68E0: 01 FE FF FF 00 01 FE FF FF 00 01 FE FF FF 00 01 01 00 1E 01 00 10 01 00   4529
68F8: 16 01 00 10 01 00 16 01 7E 7F 7F 00 01 7E 7F 7F 00 01 7E 7F 7F 00 01 7E   1934
6910: 7F 7F 00 01 01 00 1E 01 5F AB 7B 01 7F DE 7C 01 5F AB 7F 01 3F D4 7F 01   A49C
6928: 01 00 16 01 0F 2B 7F 01 7F 5B 7F 01 7F 63 7F 01 7F 63 7F 01 01 00 16 01   5008
6940: 7F EF 7F 01 FF EB 7F 01 7F FF 7F 01 15 25 04 01 01 00 1E 01 7E 7F 7F 00   2431
6958: 01 FE FF 7F 00 01 FE FF FF 00 01 FE FF FF 00 01 01 00 16 01 00 10 01 00   24A1
6970: 16 01 00 10 01 00 1E 01 7E 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 00   72B7
6988: 01 01 00 16 01 00 10 01 00 16 01 7E 7F 7F 00 01 7E 7F 7F 00 01 7E 7F 7F   74B7
69A0: 00 01 7E 7F 7F 00 01 01 00 1E 41 00 0F 02 01 00 16 01 7F D7 7F 01 5F A3   5DDF
69B8: 7C 01 3F 3C 7F 01 7F 78 7E 01 01 00 16 01 7F FF 7F 01 7F EF 7F 01 7F EB   CF5C
69D0: 7F 01 7F EB 7F 01 01 00 1E 01 7E FF FF 00 01 FE FF FF 00 01 FE FF FF 00   4B00
69E8: 01 FE FF FF 00 01 01 00 16 01 00 10 01 00 16 01 00 10 01 00 1E 01 7C 7F   5D69
6A00: 7F 00 01 7E 7F 7F 00 01 7E 7F 7F 00 01 7E 7F 7F 00 01 01 00 16 01 7E 7F   F20C
6A18: 7F 01 7F 7F 7F 01 7F 7F 67 01 7F 7F 7F 01 01 00 16 01 7F FF 7F 01 7F EF   F366
6A30: 7F 01 7F E3 7F 01 7F E3 7F 01 01 00 1E 01 5F E9 7E 01 5F DA 7E 01 5F AB   1EED
6A48: 7B 01 2F E3 7B 01 01 00 16 01 1F 22 7F 01 1F 5A 7C 01 1F 22 7C 01 1F 62   3D18
6A60: 78 01 01 00 16 01 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01 01 00   2D8A
6A78: 1E 01 00 10 01 00 16 01 00 10 01 00 16 01 00 10 01 00 1E 01 1F A2 78 01   58D9
6A90: 7F D8 7D 01 1F AA 7F 01 7F FD 7F 01 01 00 16 01 2F 2B 7F 01 7F 5B 7F 01   9766
6AA8: 7F 7B 7F 01 7F 7B 7F 01 01 00 16 01 7F F7 7F 01 FF EB 7F 01 7F FF 7F 01   F46A
6AC0: 15 25 04 01 01 00 1E 01 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01   F457
6AD8: 01 00 16 01 FE FF FF 00 01 FE FF FF 00 01 FE FF FF 00 01 FE FF FF 00 01   140C
6AF0: 01 00 16 01 00 10 01 00 1E 01 01 00 0E 01 01 00 16 01 00 10 01 00 16 01   D398
6B08: 00 10 01 00 1E 01 01 00 0E 01 01 00 16 01 7F D7 7F 01 5F EB 7D 01 3F 1D   FB52
6B20: 7E 01 7F 7A 7E 01 01 00 16 01 7F E3 7F 01 7F F7 7F 01 7F E3 7F 01 7F E3   352B
6B38: 7F 01 01 00 1E 01 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01 01 00   C999
6B50: 16 01 7C 7F 7F 00 01 FE FF FF 00 01 FE FF FF 00 01 7E 7F 7F 00 01 01 00   B70A
6B68: 16 01 00 10 01 00 1E 01 00 10 01 00 16 01 7C 7F 7F 00 01 7E 7F 7F 00 01   2567
6B80: 7E 7F 7F 00 01 7E 7F 7F 00 01 01 00 16 01 7F 7F 7F 01 7F 7F 7F 01 7F 7F   360C
6B98: 7F 01 7F 7F 7F 01 01 00 1E 01 5F E1 7E 01 1F E2 78 01 1F AA 7C 01 5F D5   78D1
6BB0: 78 01 01 00 16 01 5F 3A 7F 01 5F 52 7A 01 5F 2B 7B 01 5F 5A 7A 01 01 00   5A11
6BC8: 16 01 7F FF 7F 01 7F 67 7F 01 7F E3 7F 01 7F 47 7F 01 01 00 1E 01 00 10   16D3
6BE0: 01 00 16 01 00 10 01 00 16 01 00 10 01 00 1E 01 7F 7F 7F 01 7F 7F 7F 01   616C
6BF8: 7F 7F 7F 01 7F 7C 7F 01 01 00 16 01 6F 22 7C 01 7F 63 7F 01 7F 43 7F 01   A3C3
6C10: 7F 7B 7F 01 01 00 16 01 7F FB 7F 01 FF EB 7F 01 7F F7 7F 01 1D 25 04 01   5533
6C28: 01 00 1E 01 1F 22 7B 01 1F A2 7C 01 1F 22 7B 01 0F 22 7B 01 01 00 16 01   739D
6C40: 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01 7F FF 7F 01 01 00 16 7E 7F 7F 7F 7F   1B89
6C58: 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 00 1A 01 01                           A910
```

C TRÈS SIMPLE ! (suite)

MAIS ÇA SE COMPLIQUE...

Je dois dire que ça se passe plutôt bien, à part quelques menaces de mort de la part de personnes que je croyais mes amis et un autodafé de TREMPLIN MICRO dans les universités, le dernier article de " **C** très simple" a été si bien accueilli que je vous entraîne une fois de plus sur les chemins étroits et caillouteux de la connaissance pure.

Les variables que nous avons rencontrées jusqu'à présent étaient des objets élémentaires et isolés, or le plus inculte des Basicois de basse souche vous dira que pour travailler efficacement il a besoin de regrouper de temps en temps ces variables discrètes sous forme de listes ou de matrices qu'il va nommer "TABLEAU" et déclarer celles-ci par l'instruction DIM.

Le même sous-Basicois aimerait bien pouvoir organiser ses données d'une manière un peu plus fine que ne le permet son langage préféré : par exemple regrouper sous une même variable, un nom, un prénom, une adresse, un âge, autrement dit des objets qui, bien qu'ayant un lien commun sont de TYPES différents ou encore créer une variable qui puisse contenir à différents moments des objets de types et de tailles différents.

Mais avec Basic, comme le dit la publicité SNCF, ce n'est pas possible...

ALLELUIA ! le langage **C** peut le faire, et bien d'autres choses encore, allez, je suis gentil, commençons par les choses simples.

LES TABLEAUX

Pas de malaise, dans son aspect extérieur, rien ne ressemble plus à un tableau en langage **C** qu'un tableau en BASIC. Un tableau est défini en précisant entre crochets (pour l'APPLE français entre °§) le nombre total d'éléments qu'il doit pouvoir contenir.

Par exemple : int liste°40§;

définit un tableau dont l'identificateur (le nom) est *liste* et composé de 40 objets entiers consécutifs nommés :

liste°0§, liste°1§ ..... liste°39§

Remarquez que le premier élément du tableau est d'indice 0, le dernier élément est donc d'indice NOMBRE D'ÉLÉMENTS -1.

Ecrire liste°40§=10 par exemple conduirait à des catastrophes épouvantables... j'aime mieux ne pas y penser.

C manipule des tableaux de n'importe quel type (y compris des tableaux de tableaux) avec une rare élégance. En effet, la notion de tableau en **C** est intimement liée à celle de pointeur (voir le numéro précédent). L'identificateur de tableau (liste dans l'exemple ci-dessus) est en fait un pointeur vers le premier élément du tableau.

Je commence à voir de la fumée qui sort par les oreilles de quelques-uns et quelques-unes... Peut-être ne suis-je pas très clair ? OK je m'explique. *(suite page 14)*

Lorsque vous déclarez INT TRUC°2§
vous donnez à TRUC la valeur de l'adresse mémoire où votre compilateur a prévu de placer un tableau de deux entiers consécutifs. Vous pourrez donc utiliser :
— soit la syntaxe "TABLEAU"
— soit la syntaxe "POINTEUR"

et écrire avec autant de bonheur :
TRUC°Ø§=4; ou *TRUC=4;
ou encore
TRUC°3§=1Ø équivalent à *(TRUC+3)=1Ø

Les débutants en **C** utilisent le plus souvent la syntaxe "TABLEAU" qu'ils abandonnent progressivement au profit de la seule syntaxe "POINTEUR ". Leurs programmes deviennent alors très beaux, très rapides et totalement illisibles.

Un avantage des tableaux **C** sur les tableaux "BASIC" c'est qu'ils peuvent être initialisés très facilement au moment de leur déclaration. Alors qu'en Basic on utilise des ruses pas possibles à base de DATA READ et autres joyeusetés en forme d'itérations, il suffit en **C** de faire suivre la déclaration de tableau par un bloc contenant les valeurs d'initialisation pour faire tout le travail :

float valeurs°3§ é2.718,3.14,9è;

déclare un tableau de trois réels et initialise leurs valeurs respectives à :

valeur°Ø§=2.718, valeur°1§=3.14, valeur°2§=9

de même :

char chaîne°1Ø§ = "TOTO";

déclare un tableau de caractères de 1Ø éléments capable sans souffrance de recevoir la chaîne de caractères "TOTO".

chaîne°Ø§ ='T', chaîne°1§='O' etc...

Nous y reviendrons puisque c'est la façon dont **C** gère les chaînes de caractères.

Histoire de vous dérouiller les doigts, voici un petit programme inspiré de ma course d'ordinateurs du dernier numéro :

Afficher les nombres premiers entre 2 et 1ØØØ. Je rappelle aux oublieux qu'un nombre premier est un nombre qui n'est divisible que par 1 ou par lui-même.

L'algorithme choisi est celui du crible d'Erathostène vieux de plus de 2ØØØ ans et remis au goût du jour par l'informatique familiale.

On construit un tableau dont la valeur de chaque élément est soit 1, soit Ø selon que l'indice est premier ou non.

Remarquez que le tableau "crible" dont l'indice sera au maximum 1ØØØ doit être déclaré de 1ØØ1.

```
/* ———————— ERATOTOS ———————— */

int crible°1ØØ1§;       /*le lieu du crible !!!*/

main()
é
int i,k;                /*compteurs d'itération*/

/* Assignons la valeur 1 à tous les éléments du
tableau*/

for(i=2;i<=1ØØØ;i++) crible°i§=1;

/*
Parcourons le tableau de 2 à 1ØØØ en effectuant la tâche suivante : si l'élément d'indice i
est de valeur 1 c'est que i est premier, donc on
l'écrit et on élimine tous ses multiples en attribuant la valeur Ø aux éléments du tableau
d'indices multiples de i
*/

for (i=2;i<=1ØØØ;i++)
        é
        if (crible°i§)/*si=1 : nombre premier*/

        é
        printf("%dçn",i);/* On l'écrit*/
        /*On élimine tous ses multiples*/
        for     (k=i+i;k<=1ØØØ;k=k+i)
        crible°k§ =Ø;
        è

        è

è
```

Voilà, ça ne sert pas à grand chose, mais qu'est-ce que c'est beau ! Les plus courageux pourront s'amuser à modifier le programme en syntaxe "pointeur", les plus laborieux chercheront à l'écrire en Basic ou en Pascal, les plus impatients se rallieront à mon panache blanc afin qu'il les guide dans le dédale des abominables structures. *(suite page 14)*

LES STRUCTURES

Les tableaux, c'est bien mais ce n'est pas tout dans la vie. Il peut arriver que vous deviez représenter un objet composite constitué de grandeurs de même type ou de types différents : par exemple une date composée de trois entiers (le jour, le mois, l'année) ou encore le nom des mois de l'année (chaîne) associés à un nombre de jours (entier).

C permet de construire de tels agrégats à l'aide de la déclaration STRUCT (structure) :

```
struct  date é
        int jour;
        int mois;
        int année;
        è;
```

Les Pascaliens reconnaîtront sans peine un type composé qui leur est cher : le type RECORD ou article.

Les Basicois ne reconnaîtront rien mais c'est bien fait pour eux.

Comme je les aime bien, je vais tout de même préciser pour ces déshérités la triste vérité suivante : ce que vous venez de fabriquer laborieusement n'est pas une variable mais un TYPE d'objet que vous utiliserez ensuite pour déclarer une variable éventuellement initialisée.

Par exemple :

```
struct date anniversaire = é10.11.1987è;
```

date est un TYPE de structure, **anniversaire** (en fait le mien, à bon entendeur...) est une VARIABLE de type date.

On peut également déclarer directement une variable structure en faisant suivre une déclaration de type structure par un nom de variable, c'est à mon avis moins clair quoique plus rapide ainsi :

```
struct    é
                int jour;
                int mois;
                int année;
                è anniversaire = é10,11,1987è;
```

est équivalent du point de vue fonctionnement à la déclaration "déclaration de type + déclaration de variable" vue précédemment.

Pour avoir accès à un élément d'une structure, nous devons utiliser l'opérateur "." (point).

Par exemple : anniversaire.jour

désigne l'élément jour de la structure anniversaire.

Vous l'avez sûrement déjà compris, le pied, c'est le tableau de structures : il combine l'accès facile à un élément par un indice (tableaux) et la clarté descriptive des structures :

```
struct date datesimportantes°2§
                                é
                                10,10,1515,
                                14,7,1789,
                                10,11,1987
                                è
```

déclare un tableau de 3 structures où, par exemple, datesimportantes°0§.année est égal à 1515.

Ceci n'est pas horriblement complexe. Malheureusement, le pointeur fou a également frappé les structures et il est très courant de déclarer un pointeur vers une structure :

si on déclare :

```
struct date *p;
p = &anniversaire;
```

on disposera des syntaxes :

(*p).jour ou encore p -> jour équivalentes à date.jour

Pour l'instant, tout ceci doit vous laisser un peu froids, mais n'oubliez pas que **C** est avant tout un outil de manipulation de pointeurs : en **C** quand on peut créer un pointeur vers quelque chose, on est sauvé...

Essayez pour vous défouler le petit exemple ci-dessous :

```
struct homme
                é
                char nom °15§;
                char prénom °15§;
                int âge;
                è;
struct homme personne é"TOTO","JEAN",18è;
struct homme *p;

main()
é
```

(suite page 16)

```
printf("%sçn%sçn%d%d",homme.nom,homme.prénom,homme.âge);
p=&personne;
printf("%sçn%sçn%d%d",(*p).nom,(*p).prénom,(*p).âge);
printf("%sçn%sçn%d%d",p ->nom,p -> prénom,p ->âge);
è
```

LES UNIONS

Vu de l'extérieur une union ressemble furieusement à une structure ; la déclaration est très proche. La seule différence apparaît dans le mot "union" qui remplace le mot "struc". Mais attention, cettè ressemblance n'est qu'apparente : une union est prévue pour contenir séquentiellement des objets de types et de tailles différents, c'est-à-dire qu'à un instant donné UN SEUL PEUT ÊTRE PRÉSENT en mémoire.

On peut voir une union comme une structure dont les différents champs commencent tous à la même adresse mémoire.

```
union   é
                int i;
                char c;
                long l;
    è monunion;
```

La déclaration ci-dessus permet de créer une variable nommée monunion qui pourra recevoir à un moment donné soit un entier, soit un caractère, soit un réel.

```
monunion,i=255;
monunion.c='c';
monunion.l:=3.1416;
```

Ces trois lignes pourraient être une assignation valable, mais si vous les écrivez dans un programme, seule monuion.l sera significative puisqu'elle aura "écrasé" monunion.c qui aura elle-même écrasé monunion.i.

Le type UNION permet de construire des structures à champ variable semblables au CASE du RECORD Pascal avec plus de souplesse et plus d'efficacité mais ça commence à devenir difficile...

Le compilateur APPLE reconnaît un autre type : le type ENUM ou énumération dérivé directement du type énumération de PASCAL.

Son intérêt principal est d'ailleurs de permettre une traduction aisée de PASCAL en **C** . On peut déclarer des types du genre :

```
enum Boissons éCocaCola,Perrier,Orangina,
SevenUpè;
```

l'utilisation de ce type rend les programmes intransportables sur I.B.M. ou VAX. Je ne vous parlerai donc pas aujourd'hui de cette horreur...

REMARQUES, MESSAGES, NOTES, EXCUSES, REGRETS, ESPOIRS

1° La prochaine fois, **C** sera vraiment très simple.

2° Pour aujourd'hui, pardonnez-moi vos maux de tête, mais il fallait en passer par là.

3° La sécurité sociale ne remboursant pas l'aspirine, je ne me sens pas impliqué dans son déficit.

4° Les sociétés CocaCola, Perrier, Orangina, etc. peuvent faire parvenir leurs chèques au journal qui transmettra.

5° Mon anniversaire est bien le 10 Novembre.

6° A bientôt... **Claude AUBRY**

Si vous écrivez à Claude Aubry, n'oubliez pas de joindre une enveloppe timbrée pour la réponse. Merci !

SUR LES TABLES DE FORMES (shapes)

Yvan KŒNIG

A plusieurs reprises, *Tremplin Micro* a publié des tables de formes, mais sans en rappeler la structure. Je vais essayer de combler cette lacune. Rappelons qu'une *forme* est un ensemble codé de vecteurs de déplacement :

— certains dits actifs parce qu'ils affichent un point lumineux à leur position de départ,
— d'autres, passifs, n'affichent rien.

Il existe 8 vecteurs différents :

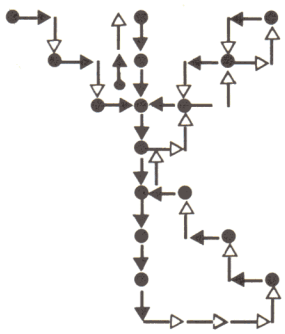
PASSIFS		ACTIFS	
déplacement vers le haut	0 0 0	déplacement vers le haut	1 0 0
déplacement vers la droite	0 0 1	déplacement vers la droite	1 0 1
déplacement vers le bas	0 1 0	déplacement vers le bas	1 1 0
déplacement vers la gauche	0 1 1	déplacement vers la gauche	1 1 1

Soit à définir, par exemple, le caractère qui me sert parfois de *signature,* à savoir :

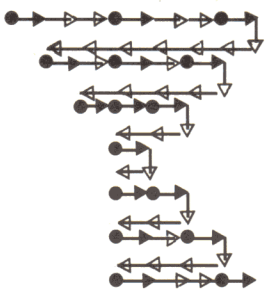
```
* . . * . . *
. * . * . * .
. . * * * . .
. . . * . . .
. . . * * . .
. . . * . * .
. . . * . . *
```

Nous allons *balayer* ce caractère de deux façons, afin de montrer qu'il n'y a pas une solution unique et surtout, de mettre en évidence l'incidence du mode retenu lorsque l'on emploie la possibilité offerte par APPLESOFT pour dessiner une forme en différentes tailles grâce à SCALE.

Balayage 1



Balayage 2



Ce qui se traduit après codage par :

0 0	0 1 0	1 0 1	$15	0 1	0 0 1	1 0 1	$4D
0 0	0 1 0	1 0 1	$15	0 1	0 0 1	1 0 1	$4D
0 0	1 0 0	1 0 1	$25	1 1	0 1 0	1 0 1	$D5
0 0	1 1 0	0 0 0	$30	1 1	0 1 1	0 1 1	$DB
0 0	1 1 0	1 1 0	$36	0 0	0 1 1	0 1 1	$1B
0 0	1 1 0	1 1 0	$36	0 0	0 0 1	1 0 1	$0D
0 0	1 1 0	1 1 0	$36	0 0	0 0 1	1 0 1	$0D
0 1	0 0 1	0 0 1	$49	1 1	0 1 0	1 0 1	$D5
0 0	1 1 1	0 0 0	$38	1 1	0 1 1	0 1 1	$DB
0 0	1 1 1	0 0 0	$38	0 0	1 0 1	1 0 1	$2D
0 0	1 1 1	0 0 0	$38	1 1	0 1 0	1 0 1	$D5
0 0	0 0 1	0 0 0	$08	1 0	1 0 1	0 1 1	$AB
0 0	0 0 1	0 0 0	$08	0 0	1 0 1	0 1 1	$2B
0 0	0 0 1	0 0 0	$08	1 1	0 1 0	1 0 1	$D5
1 0	1 1 1	0 0 0	$B8	0 1	1 0 1	0 1 1	$6B
0 0	0 1 0	1 1 1	$17	1 1	0 1 0	1 0 1	$D5
0 0	0 0 0	1 1 1	$07	0 0	0 1 1	0 1 1	$1B
terminateur			$00	0 1	0 0 1	1 0 1	$4D
				0 0	0 0 0	1 0 1	$05
				terminateur			$00

Vous avez sans doute remarqué que chaque octet est divisé en 3 groupes. Les deux de droite, comportant 3 bits, peuvent abriter n'importe quel type de vecteur. Le groupe de gauche, ne comportant que 2 bits, est d'usage réduit (certains éditeurs de formes ne l'utilisent jamais). On ne peut en fait y ranger que trois vecteurs, tous de type passif : vers la droite, le bas ou la gauche.

Une définition de forme se terminant par un 0 0, il n'est pas possible de coder deux déplacements passifs consécutifs vers le haut. *(suite page 18)*

De même, si les 5 bits de gauche sont à Ø, aucun déplacement n'est généré. C'est pour cela que, dans le balayage 1, le vecteur central du 3e octet a été codé actif, ce qui ne paraissait pas indispensable au premier abord.

Maintenant que nous avons deux définitions de formes, il nous faut les insérer dans une TABLE DE FORMES exploitable par Applesoft.

Une telle table est constituée par une succession de définitions de formes, terminées chacune par un octet nul. En tête de cet ensemble, on place un *répertoir* des formes qui permet d'accéder aux formes elles-mêmes.

Le premier octet de ce répertoire contient le nombre de formes de la table (1 à 255), le second octet inemployé est mis à zéro. Ensuite, on place des pointeurs de deux octets contenant le décalage (offset) entre l'octet Ø de la table et le premier octet de la forme correspondante. Il est prudent de prévoir quelques pointeurs vides qui permettront d'ajouter aisément quelques formes complémentaires en case de besoin.

Avec les exemples ci-dessus la table va avoir la composition suivante :

```
$3ØØ
nbre formes     Ø2 ØØ
pointeur1       Ø8 ØØ
pointeur2       1A ØØ
pointeur vide   ØØ ØØ
forme1          15 15 25 3Ø 36 36 36 49 38 38 38 Ø8 Ø8 Ø8 B8 17 Ø7 ØØ
forme2          4D 4D D5 DB 1B ØD ØD D5 DB 2D D5 AB 2B D5 6B D5 1B 4D Ø5 ØØ
BSAVE YK2,A$3ØØ,L$2E
```

Le petit programme Basic joint permet d'afficher les deux formes dans trois tailles, ce qui montre que si en taille 1 l'aspect est le même, tout change avec scale 2 ou 3.

Je mets aussi en évidence l'utilisation des switches, peu connus, qui permettent de basculer MONOCHROME «——» COULEUR.

RÉFÉRENCES

Manuel de référence APPLESOFT	A2L008F
Manuel de l'utilisateur APPLE II	Poole.McNiff.Cook Editions Radio
Création de tables de formes	D. Compère POM's 4
Fusion de tables de formes	D. Sureau POM's 9
Un éditeur de formes	T. le Tallec POM's 12
Documentation de tables de formes ..	E. Ringot POM's 13
DESIGNER/ILLUSTRATOR	deux programmes remarquables édités par NIBBLE au prix de $29.95


```
 1Ø PRINT CHR$(4) "BLOAD YK2": POKE 232,Ø: POKE 233,3
 2Ø TEXT : PRINT CHR$(21): HCOLOR=3: ROT=Ø
 3Ø HGR
 4Ø SCALE=1: DRAW 1 AT 89,1Ø: DRAW 2 AT 189,1Ø
 5Ø SCALE=2: DRAW 1 AT 86,3Ø: DRAW 2 AT 186,3Ø
 6Ø SCALE=3: DRAW 1 AT 83,6Ø: DRAW 2 AT 183,6Ø
 65 SCALE=4: DRAW 1 AT 8Ø,1ØØ: DRAW 2 AT 18Ø,1ØØ
 7Ø FOR J=1 TO 1Ø
 8Ø POKE 49246,Ø: GOSUB 2ØØ: REM  monochrome
 9Ø POKE 49247,Ø: GOSUB 2ØØ: REM  couleur
1ØØ NEXT
11Ø POKE 49246,Ø: REM  monochrome
12Ø END
2ØØ FOR I = 1 TO 1ØØØ: NEXT : RETURN
```

DATA-SUITE

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

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

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

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

Que croyez-vous qu'elle affichera ? ■

DATA-S1

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

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

```
300 : A5 67     LDA $67
302 : 85 7F     STA $7F
304 : A5 68     LDA $68
306 : 85 80     STA $80
308 : 80 31     BRA $033B
30A : A4 80     LDY $80
30C : A6 7F     LDX $7F
30E : E8        INX
30F : D0 01     BNE $0312
311 : C8        INY
312 : 98        TYA
313 : 20 24 ED  JSR $ED24
316 : 20 48 F9  JSR $F948
319 : A0 00     LDY £$00
31B : E6 7F     INC $7F
31D : D0 02     BNE $0321
31F : E6 80     INC $80
321 : B1 7F     LDA ($7F),Y
323 : F0 0B     BEQ $0330
325 : C9 2C     CMP £$2C
327 : F0 07     BEQ $0330
329 : 09 80     ORA £$80
32B : 20 ED FD  JSR $FDED
32E : 80 E9     BRA $0319
330 : 48        PHA
331 : A9 8D     LDA £$8D
333 : 20 ED FD  JSR $FDED
336 : 68        PLA
337 : C9 2C     CMP £$2C
339 : F0 CF     BEQ $030A
33B : E6 7F     INC $7F
33D : D0 02     BNE $0341
33F : E6 80     INC $80
341 : A0 00     LDY £$00
343 : B1 7F     LDA ($7F),Y
345 : F0 06     BEQ $034D
347 : C9 83     CMP £$83
349 : F0 BF     BEQ $030A
34B : D0 EE     BNE $033B
34D : A5 80     LDA $80
34F : C5 6A     CMP $6A
351 : 90 E8     BCC $033B
353 : A5 7F     LDA $7F
355 : C5 69     CMP $69
357 : 90 E2     BCC $033B
359 : 60        RTS
```

Marcel COTTINI

Les vecteurs d'interruptions sous ProDOS

LES INTERRUPTIONS POSSIBLES

NMI Non Maskable Interrupt : ce sont les interruptions non masquables.

RESET Monitor Interrupt : ce type d'interruption renvoie le système dans le Monitor. Elle est gérée à partir de l'adresse $FA62.

IRQ Interrupt Request : ce sont les interruptions masquables, dont le BReaK fait partie.

Pour chaque signal d'interruption, il existe une ligne particulière d'accès au processeur : si le signal véhiculé est à un niveau bas (Ø), le processeur reçoit une demande d'interruption ; par contre, si le signal est à un niveau haut (1), rien ne se passe.

Adresses d'interruptions :

$FFFA-$FFFB pour une interruption NMI JMP ($FFFA)
$FFFC-$FFFD pour une interruption RESET JMP ($FFFC)
$FFFE-$FFFF pour une interruption IRQ JMP ($FFFE)

Processus d'une interruption type :

— L'interruption en cours est menée à terme ;
— L'octet de poids fort du compteur ordinal est empilé — PCH ;
— L'octet de poids faible du compteur ordinal est empilé — PCL ;
— Le contenu du registre d'état PS est empilé ;
— Le bit 2, inhibition des interruptions, est mis à 1 ;
— Exécution de l'instruction qui se trouve à l'adresse $FFFE de la ROM Monitor, ou saut indirect à l'adresse contenue dans les octets $FFFF-$FFFE.

NMI INTERRUPT

ROM Monitor commutée :

La ROM du Monitor, adresses $FFFA-$FFFF, contient le pointeur du vecteur d'appel pour gérer la demande d'interruption adressée au microprocesseur. Le premier pointeur 16 bits contenu dans $FFFA-$FFFB est réservé aux interruptions du type NMI. Pour un Apple *IIe* ou *IIc*, les adresses $FFFA-$FFFB contiennent : FB Ø3 = $Ø3FB. Lors du boot d'une disquette ProDOS, l'adresse $Ø3FB contient l'adresse de saut au Monitor.

Vecteur du NMI :

```
Ø3FB – 4C 59 FF  JMP $FF59

FF59 – 2Ø 84 FE  JSR $FE84  SETNORM. Mode normal
FF5C – 2Ø 2F FB  JSR $FB2F  INIT. Mode texte
FF5F – 2Ø 93 FE  JSR $FE93  SETVID. Mode PR£Ø
FF62 – 2Ø 89 FE  JSR $FE89  SETKBD. Mode IN£Ø
FF65 – D8        CLD
FF66 – 2Ø 3A FF  JSR $FF3A  BELL. Sonnette
```

Après l'exécution de la routine implantée à partir de l'adresse $FF59, le système effectue un saut dans le Monitor, et affiche l'astérisque (*) comme prompt.

Carte langage commutée :

La copie de l'adresse $Ø3FB se trouve dans la carte langage, aux adresses $FFFA-$FFFB (FB Ø3 = $Ø3FB) pour être utilisée par le MLI lorsqu'une interruption NMI intervient. Comme la carte langage n'est pas protégée en lecture lors d'une interruption du type NMI, IRQ ou RESET, le processeur se branche à $Ø3FB, puis effectue un saut à $FF59 et exécute le sous-programme. Si le pseudo disque (RAMDisk) est utilisé avec la carte auxiliaire 64 Ko ou avec un Apple *IIc*, une interruption NMI plante le système et renvoie l'utilisateur dans le Monitor (Prompt = *). L'instruction CALL 976 (3DØG) reconnecte le Basic System en effectuant un démarrage à chaud, tandis que CALL 39447 appelle la routine Connect qui se trouve à l'adresse $9A17.

(suite page 22)

RESET INTERRUPT

ROM-Monitor commutée :

Le point d'entrée de RESET se trouve à l'adresse $FA62. La copie de cette adresse se trouve au vecteur $FFFC-$FFFD ($FFFC- 62 FA) de la ROM Monitor.

RESET avec une ROM autostart (Apple IIe)

```
FA62 - D8                  Annulation du mode décimal.
FA63 - 20 84 FE   JSR $FE84  SETNORM. Mode normal.
FA66 - 20 2F FB   JSR $FB2F  INIT. Mode texte.
FA69 - 20 93 FE   JSR $FE93  SETKBD. Mode IN£0.
FA6C - 20 89 FE   JSR $FE89  SETVID. Mode PR£0.
FA6F - AD 58 C0   LDA $C058  Annonciateur 0 à 1 — Read.
FA72 - AD 5A C0   LDA $C05A  Annonciateur 0 à 1 — Write.
FA75 - A0 05      LDY £$05   Initialise le registre Y à la valeur £$05.
FA77 - 20 B4 FB   JSR $FBB4  Gestion du curseur — avance 5 positions.
FA7B - AD FF CF   LDA $CFFF  CLRROM Désactive les sous-programmes placés entre $C800 à
                             $CFFF.
FA7E - 2C 10 C0   BIT $C010  KBDSTRB. Echantillonnage du clavier.
FA81 - D8         CLD
FA82 - 20 3A FF   JSR $FF3A  BELL Sonnette.
FA85 - AD F3 03   LDA $03F3  RESET-Handler. Charge le pointeur HByte.
FA88 - 49 A5      EOR £$A5   PRWRUP Byte.
FA8A - CD F4 03   CMP $03F4
FA8D - D0 17      BNE $FAA6  Démarrage à froid. Reboote le système.
FA8F - AD F2 03   LDA $03F2  RESET-Handler. Charge le pointeur LByte.
FA92 - D0 0F      BNE $FAA3  Si différent de 0, démarrage à chaud.
FA94 - A9 E0      LDA £$E0
FA96 - CD F3 03   CMP $03F3  Est-ce le vecteur Basic ? — $E000.
FA99 - D0 08      BNE $FAA3  Non, démarrage à chaud.
FA9B - A0 03      LDY £$03   Fixe RESET sur le vecteur Basic à froid.
FA9D - 8C F2 03   STY $03F2  Démarrage en $E003 — Basic à chaud.
FAA0 - 4C 00 E0   JMP $E000  Basic à froid — Coldstart.

FAA3 - 6C F2 03   JMP ($03F2) Warmstart. Saut au Handler.

FAA6 - 20 60 FB   JSR $FB60  Coldstart. Entrée du démarrage à froid.
```

RESET-Handler. Entry Moniteur (RESET = $FA62)

```
FFCB - AD D7 FF   LDA $FFD7  Charge le pointeur HByte,
FFCE - 48         PHA        et l'empile.
FFCF - AD D6 FF   LDA $FFD6  Charge le pointeur LByte,
FFD2 - 48         PHA        et l'empile. Adresse de retour — $FA61 —.
FFD3 - 4C C8 FF   JMP $FFC8  Commute la ROM moniteur.
FFD6 - 61 FA                 Pointeur $FA61 — 61 FA —.
```

Processus : A l'adresse $FFCB du Monitor se trouve un RTS (code 60). Conséquence : le pointeur contenu aux adresses $FFD6-$FFD7 sera utilisé par le processeur pour effectuer un retour à l'adresse $FA62. L'adresse de retour dépilée après une instruction RTS, sera augmentée de 1: $FA61 + £$01 = $FA62. C'est une caractéristique propre aux microprocesseurs 6502 et 65C02.

IRQ INTERRUPT

Cette interruption n'est autorisée par le microprocesseur, que si le bit d'inhibition, bit 2 du registre d'état PS, est à zéro. Cette situation est provoquée par l'instruction CLI du processeur.

IRQ Interrupt pour un Apple IIe

```
FA40 - 85 45     STA $45      Sauvegarde de l'accumulateur.
FA42 - 68        PLA          Dépile le registre d'état PS.
FA43 - 0A        ASL          Bit 4 positionné à 1, si IRQ est transmis par une instruction
FA44 - 0A        ASL          BReaK — 00 —.
FA47 - 30 03     BMI $FA4C    BReaK avec affichage des registres.
FA49 - 6C FE 03  JMP ($03FE)  Adresse IRQ-Handler.
```

ProDOS positionne IRQ-Handler à la valeur $BFEB

```
03FE - EB                     LByte du pointeur IRQ-Handler.
03FF - BF                     HByte du pointeur IRQ-Handler.

BFEB - AD 8B C0  LDA $C08B    BankSelect. Bank 1 autorisation de lecture,
BFEE - AD 8B C0  LDA $C08B    + écriture.
BFF1 - 4C 3A D1  JMP $D13A    IRQ-Handler à partir du MLI.
```

IRQ-HANDLER ENTRY (MLI)

```
D13A - A5 45     LDA $45      Charge la valeur de l'adresse $45 dans l'accumulateur —
                              Registre A —.
D13C - 8D 88 BF  STA $BF88    INTAREG. Sauvegarde de l'accumulateur.
D13F - 8E 89 BF  STX $BF89    INTXREG. Sauvegarde de X dans la page globale.
D142 - 8C 8A BF  STY $BF8A    INTYREG. Sauvegarde de Y dans la page globale.
D145 - BA        TSX          Copie du pointeur de pile dans X.
D146 - 8E 8B BF  STX $BF8B    INTSREG. Sauvegarde du pointeur de pile.
D149 - 68        PLA          Dépile la valeur du registre d'état PS.
D14A - 8D 8C BF  STA $BF8C    INTPREG. Sauvegarde du registre d'état PS.
D14D - 68        PLA          Dépile l'adresse de retour ; LByte.
D14E - 8D 8E BF  STA $BF8E    INTADR. Sauvegarde de l'adresse de retour ; LByte.
D151 - 68        PLA          Dépile l'adresse de retour ; HByte.
D152 - 8D 8F BF  STA $BF8F    Sauvegarde de l'adresse de retour ; HByte.
D155 - 9A        TXS          Positionne le pointeur de pile.
D156 - AD F8 07  LDA $07F8    MSLOT. Numéro de l'interface ; HByte.
D159 - 8D C4 D1  STA $D1C4    Sauvegarde du numéro de l'interface.
D15C - BA        TSX
D15D - 30 09     BMI $D168    Saut, si le pointeur de pile > £$7F, sinon,
D15F - A0 0F     LDY £$0F     initialise le registre Y à la valeur 16,
D161 - 68        PLA          et dépile 16 valeurs.
D162 - 99 AF F0  STA $F0AF,Y  Sauvegarde des valeurs dépilées.
D165 - 88        DEY          Décrémente Y d'une unité,
D166 - 10 F9     BPL $D161    et continue à dépiler.
D168 - A2 FA     LDX £$FA
D16A - B5 00     LDA $00,X    Sauvegarde des valeurs de la page zéro,
```

(suite page 24)

```
D16C - 9D A5 EF    STA $EFA5,X   de £$FA à £$FF.
D16F - E8          INX           Incrémente la registre X,
D170 - D0 F8       BNE $D16A     et continue avec la prochaine valeur.
```

INTERRUPTS VECTORS

```
D172 - AD 81 BF    LDA $BF81     Interrupt 1 : charge HByte de l'adresse de saut,
D175 - F0 05       BEQ $D17C     adresse vide, charge le vecteur suivant.
D177 - 20 CE D1    JSR $D1CE     Vecteur utilisé : appel du sous-programme.
D17A - 90 23       BCC $D19F     Restaure la page zéro.

D17C - AD 83 BF    LDA $BF83     Interrupt 2 : charge HByte de l'adresse de saut,
D17F - F0 05       BEQ $D186     adresse vide, charge le vecteur suivant.
D181 - 20 D1 D1    JSR $D1D1     Vecteur utilisé : appel du sous-programme.
D184 - 90 19       BCC $D19F     Restaure la page zéro.

D186 - AD 85 BF    LDA $BF85     Interrupt 3 : charge HByte de l'adresse de saut,
D189 - F0 05       BEQ $D190     adresse vide, charge le vecteur suivant.
D18B - 20 D4 D1    JSR $D1D4     Vecteur utilisé : appel du sous-programme.
D18E - 90 0F       BCC $D19F     Restaure la page zéro.

D190 - AD 87 BF    LDA $BF87     Interrupt 4 : charge HByte de l'adresse de saut,
D193 - F0 05       BEQ $D19A     adresse vide, charge le vecteur suivant.
D195 - 20 D7 D1    JSR $D1D7     Vecteur utilisé : appel du sous-programme.
D198 - 90 05       BCC $D19F     Restaure la page zéro.
```

ERROR HANDLER

```
D19A - A9 01       LDA ££$01     Aucun des pointeurs n'est en place dans la page globale de
                                 PRODOS, aux adresses $BF80-$BF87.
                                 Charge l'accumulateur avec la valeur £$01 qui sera utilisée
                                 comme drapeau — Flag — pour le code d'erreur.
D19C - 20 0C BF    JSR $BF0C     SYSTDEATH. Appel de la routine d'erreur : contient normale-
                                 ment un saut à $D1E6.
```

Restaure la page zéro

```
D19F - A2 FA       LDX £$FA      Initialise le registre X comme compteur.
D1A1 - BD A5 EF    LDA $EFA5,X   Charge les valeurs des adresses indexées,
D1A4 - 95 00       STA $00,X     et les sauvegarde dans la page zéro.
D1A6 - E8          INX           Incrémente de 1 le registre X,
D1A7 - D0 F8       BNE $D1A1     et continue la sauvegarde.
D1A9 - AE 8B BF    LDX $BF8B     INTSREG. Charge le pointeur de pile,
D1AC - 30 0B       BMI $D1D9     y a-t-il assez de place ? oui
D1AE - A0 00       LDY £$00      sinon, empile les valeurs sauvegardées
                                 au préalable : voir à partir de $D161.
D1B0 - B9 AF F0    LDA $F0AF,Y   Charge les valeurs des adresses indexées,
D1B3 - 48          PHA           et les empile.
D1B4 - C8          INY           Incrémente de 1 le registre Y,
D1B5 - C0 10       CPY £$10      et compare si les 16 valeurs ont été sauvegardées.
D1B7 - D0 F7       BNE $D1B0     Continue d'empiler.
```

```
D1B9 - AC 8A BF    LDY $BF8A   INTYREG. Charge la valeur du registre Y.
D1BC - AE 89 BF    LDX $BF89   INTXREG. Charge la valeur du registre X.
D1BF - AD FF CF    LDA $CFFF   CLRROM. Déconnecte les sous-programmes des ROM, entre
                               $C800-$CFFF.
D1C2 - AD 00 C1    LDA $C100   REACTIV CARD. Charge le numéro du périphérique
D1C5 - AD C4 D1    LDA $D1C4   HByte — $D1C4 de l'adresse $D159 —
D1C8 - 8D F8 07    STA $07F8   et restaure MSLOT — $07F8 —.
D1CB - 4C D0 BF    JMP $BFD0   INTEXIT. Sortie de la routine via la "globale page" de PRODOS.
```

Appel des sous-programmes, suite à une interruption :

Ces sous-programmes sont uniquement sollicités par un "JSR" à partir des adresses $D177 ; $D181 ; $D18B et $D195.

```
D1CE - 6C 80 BF    JMP ($BF80)  Exécution de la routine Interrupt 1
D1D1 - 6C 82 BF    JMP ($BF82)  Exécution de la routine Interrupt 2
D1D4 - 6C 84 BF    JMP ($BF84)  Exécution de la routine Interrupt 3
D1D7 - 6C 86 BF    JMP ($BF86)  Exécution de la routine Interrupt 4
```

SET SYSERR Entrée par un JMP venant de l'adresse $BF09.

```
D1DA - 8D 0F BF    STA $BF0F   Sauvegarde du code d'erreur.
D1DD - 68          PLA         Dépile l'adresse de retour ; LByte
D1DE - 68          PLA         HByte.
D1DF - AD 0F BF    LDA $BF0F   Charge le code de l'erreur
D1E2 - 38          SEC         Utilisé comme indicateur de l'erreur.
D1E3 - 60          RTS         Retour : normalement à $D078.
```

System DEATH Entry 2 :

L'entrée du vecteur $D1E4 est appelée par la page globale de PRODOS, à savoir : REBOOT ($BF03) et SYSDEATH2 ($BFF6).

```
D1E4 - A9 00       LDA £$00    Fixe la valeur £$00 au code d'erreur.
```

System DEATH Entry 1 :

A l'entrée de la routine, l'accumulateur contient le code d'erreur rencontré. L'appel vient de la page globale de PRODOS, à partir de l'adresse $BF0C.

```
D1E6 - AA          TAX         Copie le code d'erreur dans X.
D1E7 - 8D 0C C0    STA $C00C   Déconnecte la carte 80 colonnes — IIe et IIc.
D1EA - AD 51 C0    LDA $C051   Place le système en mode texte,
D1ED - AD 53 C0    LDA $C053   pleine page,
D1F0 - AD 54 C0    LDA $C054   page 1,
D1F3 - AD 56 C0    LDA $C056   et en basse résolution — LORES.
D1F6 - A0 27       LDY £$27    Initialise le registre Y, comme compteur
D1F8 - A9 A0       LDA £$A0    et charge la valeur du nombre d'espaces.
D1FA - 99 50 07    STA $0750,Y Affiche à l'écran le texte : INSERT SYSTEM DISK AND RESTART.
D1FD - B9 DE EF    LDA $EFDE,Y Charge les caractères stockés dans la carte langage,
D200 - 99 D0 07    STA $07D0,Y continue l'affichage des caractères à l'écran,
D203 - 88          DEY         décrémente d'une unité le compteur Y,
D204 - 10 F2       BPL $D1F8   et continue l'affichage.
D206 - 8A          TXA         Recopie le code d'erreur dans A,
D207 - F0 30       BEQ $D239   si 00, le code ne sera pas affiché : -ERRxx.
D209 - A9 AD       LDA £$AD    Charge le code ASCII de "-",
D20B - 8D F1 07    STA $07F1   et l'affiche à l'écran.
D20E - A9 C5       LDA £$C5    Charge le code ASCII de "E",
```

(suite page 26)

```
D210 - 8D F2 07    STA $07F2     et l'affiche à l'écran.
D213 - A9 D2       LDA £$D2      Charge le code ASCII de "R",
D215 - 8D F3 07    STA $07F3     et l'affiche deux fois à l'écran
D218 - 8D F4 07    STA $07F4     ——» R
                                 ——» -ERR
D21B - 8A          TXA           Code vers l'accumulateur,
D21C - 4A          LSR           et décale les bits, 4 fois vers la droite,
D21D - 4A          LSR           ce qui équivaut à ne garder que les 4 bits
D21E - 4A          LSR           de poids fort.
D21F - 4A          LSR
D220 - 09 B0       ORA £$B0      Effectue un OU Inclusif avec £$B0 (176),
D222 - C9 BA       CMP £$BA      et compare si le résultat est supérieur à 9.
D224 - 90 02       BCC $D228
D226 - 69 06       ADC £$06      Pour les valeurs supérieures à 9 ——» "A" - "F",
D228 - 8B F6 07    STA $07F6     affiche la valeur correcte à l'écran.
D22B - 8A          TXA
D22C - 29 0F       AND £$0F      Valeur des 4 bits de faible poids.
D22E - 09 B0       ORA £$B0
D230 - C9 BA       CMP £$BA
D232 - 90 02       BCC $D236
D234 - 69 06       ADC £$06      Correction pour les valeurs supérieures à 9, et affichage du
D236 - 8D F7 07    STA $07F7     résultat correct.
D239 - 4C 39 D2    JMP $D239     Attente d'un RESET, car le programme boucle sur lui-même —
                                 JMP $D239.
```

INTEXIT

Sortie de la routine de gestion d'interruption, lorsque ProDOS trouve un pointeur de sous-programme aux adresses $BF80-$BF87.

```
BFD0 - AD 8D BF    LDA $BF8D     INTBANK1. Charge le statut de la ROM.
BFD3 - F0 0D       BEQ $BFE2     £$00 = Bank 1 connectée : elle le reste.
BFD5 - 30 08       BMI $BFDF     £$FF = Bank 2 à commuter.
BFD7 - 4A          LSR
BFD8 - 90 0D       BCC $BFE7     Sans signification, valeur aléatoire.
BFDA - AD 82 C0    LDA $C082     £$01 = ROMON ; IRQ à partir du moniteur.
BFDD - B0 08       BCS $BFE7     Sans signification, valeur aléatoire.
BFDF - AD 83 C0    LDA $C083     Commute la Bank 2 (protégée contre l'écriture).
BFE2 - A9 01       LDA £$01      Initialise l'accumulateur à la valeur £$01,
BFE4 - 8D 8D BF    STA $BF8D     et fixe le mode de la ROM — ROM ou Moniteur —.
BFE7 - AD 88 BF    LDA $BF88     Recharge la valeur de l'accumulateur à partir de INTAREG.
BFEA - 40          RTI           Retour de l'interruption.
```

IRQ-Handler Carte langage commutée :

```
FF9B - 48          PHA           Sauvegarde de l'accumulateur.
FF9C - A5 45       LDA $45       L'accumulateur contient le pointeur RWTB.
FF9E - 8D 56 BF    STA $BF56     INTACC. Sauvegarde du pointeur dans la page globale de PRODOS.
FFA1 - 68          PLA           Dépile vers A, la valeur initiale, contenue au moment de l'interrup-
                                 tion IRQ,
FFA2 - 85 45       STA $45       et la sauvegarde.
FFA4 - 68          PLA
FFA5 - 48          PHA           Sauvegarde du registre d'état PS.
FFA6 - 29 10       AND £$10      Teste si IRQ provient d'un BReaK,
```

```
FFA8 - D0 18      BNE $FFC2   oui, alors branche à la routine BReaK.
FFAA - AD 00 D0   LDA $D000   Sinon, teste quelle Bank est commutée.
FFAD - 49 D8      EOR £$D8    BANKID pour la Bank 1.
FFAF - F0 02      BEQ $FFB3
FFB1 - A9 FF      LDA £$FF
FFB3 - 8D 8D BF   STA $BF8D   INBANK1. £$00 Bank 1, ou £$FF Bank 2,
FFB6 - 8D 57 BF   STA $BF57   sauvegarde :
                              $BF56-$BF57 ——» $45 pour LByte
                                          ——» £$00 ou £$FF pour HByte.
                              Pointeurs possibles :
                              HByte, LByte.
                              $FF, contenu de $45.
                              $00, contenu de $45.
FFB9 - A9 BF      LDA £$BF    Charge le pointeur HByte ;
FFBB - 48         PHA         et le place au sommet de la pile.
FFBC - A9 50      LDA £$50    Charge le pointeur LByte ;
FFBE - 48         PHA         et le place au sommet de la pile. Fixe l'adresse $BF50 comme
                              adresse RTI.
FFBF - A9 04      LDA £$04    Charge le drapeau du bit d'inhibition (bit 2) du registre d'état PS
                              — IRQ masqué —,
FFC1 - 48         PHA         et le sauvegarde au sommet de la pile.
FFC2 - A9 FA      LDA £$FA    Charge le pointeur HByte,
FFC4 - 48         PHA         et le sauvegarde au sommet de la pile.
FFC5 - A9 41      LDA £$41    Charge le pointeur LByte,
FFC7 - 48         PHA         et le sauvegarde au sommet de la pile.
                              Moniteur BReaK, adresse de retour :
                              ——» $FA41 + £$01 = $FA42 (à cause du RTS : + 1).
FFC8 - 8D 82 C0   STA $C082   ROMON. Contient un RTS lorsque l'appel provient du moniteur.
```

LANGIRQ

Entrée de la routine, après une interruption IRQ appelée depuis la carte langage, et à la suite d'une instruction RTI détectée à l'adresse $BFEA de la page globale de PRODOS.

```
BF50 - 8D 8B C0   STA $C08B   BANKSELECT. Bank 1 : protégée contre l'écriture.
BF53 - 4C D8 FF   JMP $FFD8   Saut dans la carte langage.
```

LANGINTEXIT

Entrée de la carte langage : provient d'une routine sollicitée par une interruption IRQ, avec comme origine la page globale de PRODOS, après la sélection en lecture ; protégée contre l'écriture.

```
FFD8 - 8D 88 BF   STA $BF88   Sauvegarde du registre Accumulateur.
FFDB - AD 56 BF   LDA $BF56   INTACCU. Charge la valeur,
FFDE - 85 45      STA $45     et sauvegarde en A5H.
FFE0 - AD 8B C0   LDA $C08B   BANKSELECT. Commute la Bank 1
FFE3 - AD 8B C0   LDA $C08B   + écriture autorisée.
FFE6 - AD 57 BF   LDA $BF57   £$00 = Bank 1 ; £$FF = Bank 2.
                              Utilisé comme pointeur HByte.
FFE9 - 4C D3 BF   JMP $BFD3   INTEXIT2. Déconnecte la Bank et actualise RTI.
```

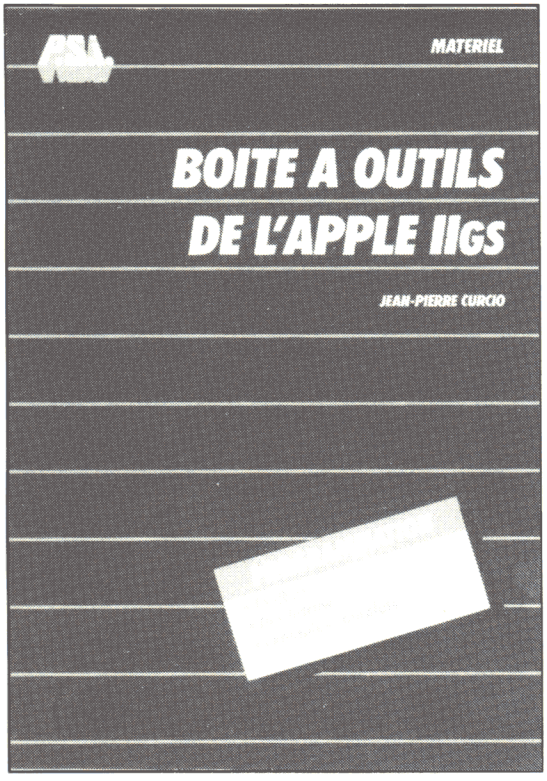
(suite page 28)

IRQ Interrupt Langage carte commutée :

```
FF9B - 48       PHA       Contenu de l'accumulateur, au moment de l'interruption IRQ.
FF9C - A5 45    LDA $45   Charge A5H, qui contient le pointeur : Read - Write — Tracks —
                          Blocks — HByte —.
FF9E - 8D 56 BF STA $BF56 Sauvegarde dans INTACCU — Page globale —.
FFA1 - 68       PLA       Dépile le contenu de l'accumulateur, au moment de l'interrup-
                          tion IRQ,
FFA2 - 85 45    STA $45   et le sauvegarde dans A5H.

FFB3 - 8D 8D BF STA $BF8D INTBANK1
FFB6 - 8D 57 BF STA $BF57 INTBANK2
```

Indispensable pour programmer l'Apple IIGS



Editions du P.S.I.

5, Place du Colonel-Fabien 75491 PARIS Cédex 10

Service Minitel :

Sur le 3615, taper OI, puis PSI

(344 pages — Reliure spirale).

On ne peut pas dire que la programmation de l'Apple *IIGS* soit d'une réelle simplicité. Bien sûr, si l'on se contente du bon vieil Applesoft, pas de problème, mais il en va autrement dès que le programmeur en herbe tente d'utiliser les nombreux et merveilleux outils du GS.

Comme le précise Jean-Pierre Curcio dans l'introduction de sa *BOÎTE À OUTILS DE L'APPLE IIGS,* la remarquable facilité qu'apporte l'interface utilisateur est liée à des règles auxquelles doit impérativement se plier le programmeur. ***Plus de facilité pour l'utilisateur signifie en l'occurrence moins de liberté pour le développeur.***

Avec l'Apple *IIGS,* on dispose d'une centaine de sous-programmes performants... mais qu'il faut apprendre à utiliser. On sait que la Toolbox de l'Apple *IIGS* ressemble à celle du Macintosh... mais ce n'est pas un avantage pour les inconditionnels de l'Apple *II.*

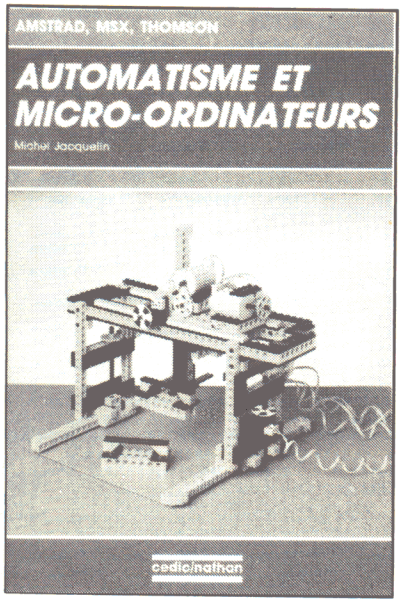
Jean-Pierre Curcio apporte une très importante contribution à la programmation avancée sur Apple *IIGS.* Chacun des chapitres de son ouvrage présente un outil (ou manager) et les routines qui le composent. Celles-ci sont illustrées par de petits exemples en **C** (d'où l'intérêt de commencer ou de poursuivre l'apprentissage de ce langage avec notre ami Claude Aubry).

A noter qu'un exemple complet est proposé à la fin de chaque chapitre et qu'une disquette d'accompagnement peut être obtenue auprès de l'auteur.

NESTOR.

Vient de paraître

Les Editeurs nous communiquent :



• AUTOMATISMES ET MICRO-ORDINATEURS (Michel Jacquelin)

La micro-informatique va de plus en plus participer à la gestion programmée des automates. Cet ouvrage s'adresse à l'utilisateur d'un micro-ordinateur (AMSTRAD, MSX, TO7-70...) ayant déjà une certaine pratique de la programmation, connaissant les matériels éducatifs LEGO (et plus particulièrement les boîtes techniques auxquelles sont généralement associés moteurs, capteurs et transmissions mécaniques diverses), et désirant réaliser lui-même des automates ou jouets programmables. Il découvrira avec ce livre, comment réaliser une machine à laver, un convoyeur, des feux tricolores ou encore un pont roulant, à partir du bus d'extension de son micro-ordinateur.

Le livre s'articule autour de cinq parties :

— La logique et le GRAFCET (logique combinatoire, séquentielle...).
— La micro-informatique (l'ordinateur, le bus d'extension...).
— Les composants de l'automate (actionneurs, capteurs).
— L'automate (définition, programmation, coupleurs et interfaces).
— L'exploitation de l'automate (langage machine, langages évoluées, BASIC).

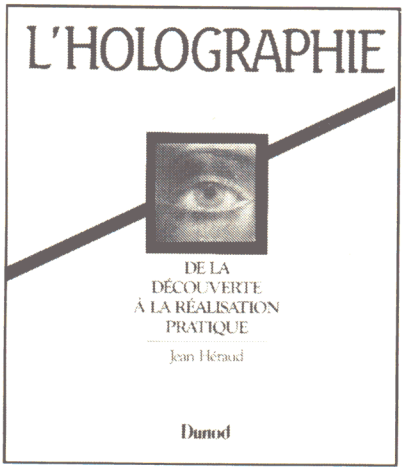
De plus, certains concepts informatiques sont rappelés au lecteur, afin de lui permettre une meilleure compréhension des contenus de certains chapitres.

cedic/nathan — *6-10, boulevard Jourdan 75014 PARIS (160 pages — 99 F).*

• L'HOLOGRAPHIE De la découverte à la réalisation pratique (Jean Héraud)

L'holographie est l'une des inventions les plus passionnantes de la science moderne, qui valut d'ailleurs à son découvreur, Denis Gabor, un juste prix Nobel en 1971. Elle permet l'élaboration d'un type nouveau de document dont le réalisme est total, les hologrammes, ces fameuses images non seulement en relief mais qui reproduisent les reflets, la micro-structure et même les vraies couleurs. Grâce à des lasers de faible puissance et peu coûteux, il est aujourd'hui aussi facile de créer des hologrammes que de réaliser des photographies. L'ouvrage de Jean Héraud permet d'aborder la technique de l'holographie, d'en comprendre les principes (optique, lumière cohérente du laser) et de passer très rapidement à la réalisation.

Livre d'initiation à la pratique, il décrit en détail, à l'aide de nombreuses figures et photographies, l'installation d'un petit laboratoire holographique : son matériel, sa mise en œuvre... Grâce aux progrès des lasers, l'imagerie holographique peut être abordée par tous. Ce livre est, en effet, destiné à tous les amateurs qui désirent aborder l'holographie, quel que soit leur niveau de connaissances (jeunes, membres de clubs, lycéens...). De plus, l'enthousiasme de l'auteur et sa grande compétence dans le domaine du laboratoire, acquise après de longues années de pratique, lui ont permis de perfectionner et de simplifier cette technique pour la mettre à la portée des non-initiés, en particulier en communiquant son savoir-faire et son expérience, si précieux pour réussir les différentes sortes d'hologrammes (par transmission, par réflexion...). Comme il le dit lui-même, "outre le côté artistique, c'est surtout la minutie de l'opérateur qui est déterminante pour la réussite des beaux hologrammes". En résumé, Jean Héraud a voulu, par cet ouvrage, permettre à ceux qui veulent aborder l'holographie de réaliser rapidement et facilement des hologrammes, après en avoir bien compris le mécanisme, afin d'en exploiter au maximum toutes les possibilités.



Dunod — *17, rue Rémy-Dumoncel B.P. 50, 75661 PARIS Cedex 14 (132 pages, broché, 120 F)*

• LES NOUVELLES LOGIQUES POUR L'INTELLIGENCE ARTIFICIELLE (Arnold Kaufmann)

L'informatique est née de l'application de la logique booléenne. Ses derniers développements à travers l'intelligence artificielle et les systèmes experts demandent de nouveaux outils qui sachent traiter l'incertain, le subjectif, l'imprécis, propres aux opérations habituelles de la pensée. *Nouvelles logiques pour l'intelligence artificielle* se propose d'étudier ces instruments. Destiné à un large public de scientifiques, d'informaticiens, mais aussi d'ingénieurs, de médecins, de biologistes, etc., cet ouvrage s'appuie sur de très nombreux exemples pour présenter les concepts caractéristiques des logiques floues, sous-ensembles aléatoires flous, représentations qui conservent la variété des jugements et reconnaissent des niveaux à la connaissance. Arnold KAUFMANN est un ancien professeur de l'Institut polytechnique de Grenoble, de l'Ecole supérieure des mines de Paris et de l'Université de Louvain. Il est l'auteur ou le co-auteur de plus de soixante ouvrages, dont certains ont été traduits en plus de vingt langues ; ces ouvrages concernent les mathématiques appliquées, la recherche opérationnelle, l'informatique, la sociologie, les méthodes de créativité, divers domaines des sciences économiques. Il convient de signaler, en particulier, ses ouvrages concernant les mathématiques floues, lesquels sont connus dans le monde entier. Les plus récents travaux d'Arnold KAUFMANN se rapportent aux logiques floues, à l'intelligence artificielle, aux systèmes experts. Il a reçu de nombreuses distinctions en France et dans le monde pour ses travaux.

Relié — 300 pages — 290 F TTC.

Editions HERMES — *51 rue Rennequin 75017 PARIS*

LOGIQUE élémentaire

Ce programme a été réalisé au MICROTEL DE MOURENX, sur APPLE *IIe* avec carte 8Ø colonnes, sous ProDOS. Comme le précise M. HUGUET, cet "exercice de style" ne sert rigoureusement à rien, sinon à amuser la galerie en décortiquant un raisonnement logique dans un univers très restreint. Il a en outre l'avantage d'exprimer son raisonnement en langage clair (nous en fournissons un exemple, directement sorti sur imprimante, à la suite du programme).

Nous le publions parce qu'il constitue un bon exercice, pour celles et ceux qui ne sont pas rompus à la pratique de l'Applesoft. Il est abondamment documenté (lire toutes les lignes de REM). De plus, il peut servir de schéma de départ pour l'élaboration de raisonnements beaucoup plus complexes... ce que nous souhaitons, bien entendu.

```
100 REM MICROTEL CLUB DE MOUREN
    XctJctJ
110 REM ACHATSctJctJ
120 REM 08 12 85     19 11 86ct
    JctJ
130 REM PROGRAMME D'INTELLIGENC
    E ARTIFICIELLE NO 1ctJctJ
140 TEXT : HOME                          1C5A
150 DIM A$(200,20),NA(200),P$(2
    00)                                  9D7D
160 D$ =  CHR$ (4)                       B3A4
170 PRINT D$;"PREFIX /MICROTEL"          531D
180 HOME : PRINT "                       2740
    """/MICROTEL/ACHATS"
190 PRINT : PRINT : PRINT "              D4B8
    "CE PROGRAMME RESOUT PAR LE
     RAISONNEMENT LE PROBLEME P
    OSE PAR UNE BASE DE ""DONNE
    ES COMPRENANT DES PHRASES D
    U TYPE:"
200 PRINT : PRINT : PRINT "              7DA5
    "TYPE: "; CHR$ (34);"DESIRE
     OBJET"; CHR$ (34)
210 PRINT : PRINT "= sujet ""DE          2A4A
    SIRE ""complement"
220 PRINT : PRINT " "PIERRE "DE          0E36
    SIRE ""DU PAIN"
230 VTAB 22: HTAB 1: PRINT "TAP          766D
    EZ UNE TOUCHE QUELCONQUE";:
    GET A$
240 HOME : PRINT "OU BIEN:"              66EB
250 PRINT : PRINT : PRINT "              04A5
    "TYPE: "; CHR$ (34);"DESIRE
     VERBE"; CHR$ (34)
260 PRINT : PRINT "= sujet ""DE          DB26
    SIRE ""complement verbal"
270 PRINT : PRINT " "JEAN """DE          4B13
    SIRE ""FAIRE DU SKI"
280 PRINT : PRINT : PRINT "AVEC          4E12
     L'EXPLICATION:"
```

```
290 PRINT : PRINT "= POUR """ve
    rbe """IL FAUT ""complement
     1 , complement 2 ET comple
    ment 3"                          0665
300 PRINT : PRINT "= POUR FAIRE
     DU SKI IL FAUT "DES SKIS "
    """, DES CHAUSSURES "ET DES
     TICKETS"                        7551
310 VTAB 22: HTAB 1: PRINT "TAP
    EZ UNE TOUCHE QUELCONQUE";:
    GET A$                           766D
320 HOME : PRINT "OU DES PHRASE
    S DU TYPE:"                      18FA
330 PRINT : PRINT : PRINT " """
    "TYPE: "; CHR$ (34);"POSSED
    E"; CHR$ (34)                    6568
340 PRINT : PRINT "= sujet ""PO
    SSEDE "complement"               2081
350 PRINT : PRINT " "JULES ""PO
    SSEDE "DES SKIS"                 279E
360 PRINT : PRINT : PRINT "AVEC
     SA COROLLAIRE:"                 18FF
370 PRINT : PRINT "= sujet ""VE
    UT VENDRE ""complement"          51B6
380 PRINT : PRINT " "JULES ""VE
    UT VENDRE ""DES SKIS"            30D3
390 VTAB 20: HTAB 1: PRINT "(L'
    ABSENCE DE COROLLAIRE SUPPO
    SE QUE JULES NE VEUT PAS VE
    NDRE DES SKIS.)"                 6A27
400 VTAB 22: HTAB 1: PRINT "TAP
    EZ UNE TOUCHE QUELCONQUE";:
    GET A$                           766D
410 GOTO 520                         0942
420 REM ctJctJRESTITUTION DES P
    HRASESctJctJctJ
430 HOME                             2F97
440 FOR I = 1 TO IJ                  2F1F
450 PRINT I;" "";                    3A1D
460 FOR T = 1 TO NA(I)               7DC0
470 PRINT A$(I,T);" ";               4313
480 NEXT T                           A7D6
490 PRINT : PRINT                    15AE
500 NEXT I                           9DCB
510 RETURN                           63B1
520 REM ctJctJ MENUctJctJ
530 HOME : VTAB 10: PRINT "VOUL
    EZ-VOUS TRAITER UNE ANCIENN
    E BASE DE DONNEES? """TAPE
    Z A"                             FCE3
540 PRINT : PRINT "OU BIEN COMP
    LETER UNE ANCIENNE BASE DE
    DONNEES? """""TAPEZ M"           9C28
550 PRINT : PRINT "OU ENCORE EN
     CREER UNE NOUVELLE? """""
    """""""""""""""TAPEZ C"          A32E
560 GET A$: IF A$ =  CHR$ (27)
    THEN END                         6224
570 IF A$ = "A" THEN 1630            35F5
580 IF A$ = "M" THEN 600             3CCD
590 IF A$ <  > "C" THEN 530          DB95
600 REM ctJctJNOM?ctJctJ
610 PRINT D$;"CATALOG"               789C
620 INPUT "NOM DE LA BASE DE DO
    NNEES DESIREE ? ";R$             A969
630 REM ctJctJOUVERTUREctJctJct
    J
640 ONERR  GOTO 780                  7EEF
650 PRINT D$;"OPEN";R$               8984
660 PRINT D$;"READ";R$               D26E
670 INPUT IJ                         1117
680 FOR I = 1 TO IJ                  2F1F
690 INPUT NA(I)                      AAAD
700 FOR T = 1 TO NA(I)               7DC0
710 INPUT A$(I,T)                    CF03
720 NEXT T                           A7D6
730 NEXT I                           9DCB
740 PRINT D$;"CLOSE"                 C517
750 POKE 216,0                       0CAE
760 GOSUB 420                        F946
770 PRINT : PRINT "TAPEZ UNE TO
    UCHE QUELCONQUE.": GET A$:
    GOTO 860                         3996
780 HOME : PRINT "IL SEMBLE QUE
     CETTE BASE DE DONNEES NE C
    ONVIENNE PAS. VOULEZ-VOUS R
    ECOMMENCER?"                     53A4
790 POKE 216,0                       0CAE
800 GET A$: PRINT A$: IF A$ =
    CHR$ (27) THEN 2910              F8C9
810 IF A$ = "O" THEN 600             42CF
820 IF A$ = "N" THEN 2910            F804
830 GOTO 780                         294A
840 REM ctJctJENTREE DE DONNEES
    ctJctJ
850 I = 0                            6549
860 I = I + 1:T = 1:K = 1:P$(I)
    = ""                             DDCC
870 REM ctJINSTRUCTIONSctJ
880 HOME : PRINT "PHRASES DU TY
    PE:": PRINT " "X "DESIRE "Y
    "                                D19F
890 PRINT " "POUR "Y "IL FAUT Z
    1, Z2,... ET ZN"                 72FC
900 PRINT " "X "POSSEDE "Y": PR
    INT " "X "VEUT VENDRE "Y"        4753
910 PRINT "(L'ABSENCE DE MENTIO
    N: X VEUT VENDRE Y SUPPOSE
```

```
    QUE X NE VEUT PAS VENDRE Y)
    "                                3053
 920 PRINT "*** ¨ANNULE LA PHRAS
    E PRECEDENTE"                    700E
 930 PRINT "FIN ¨APRES LA DERNIE
    RE PHRASE"                       BB94
 940 REM ctJENTREE DES CARACTERE
    SctJ
 950 VTAB 9: HTAB 1: PRINT I;" "
    ;                                BCF3
 960 GET C$: PRINT C$: IF C$ =
    CHR$ (44) THEN C$ =  CHR$ (
    42)                              6BD7
 970 IF C$ =  CHR$ (13) THEN 104
    0                                DB09
 980 IF C$ =  CHR$ (8) THEN  IF
    LEN (P$(I)) = 0 THEN  PRINT
    CHR$ (7): GOTO 960               4678
 990 IF C$ =  CHR$ (8) THEN  IF
    LEN (P$(I)) = 1 THEN P$(I)
    = "": GOTO 950                   A771
1000 IF C$ =  CHR$ (8) THEN P$(I
    ) =  LEFT$ (P$(I), LEN (P$(
    I)) - 1): GOTO 1020              C04D
1010 P$(I) = P$(I) + C$              A51B
1020 VTAB 9: HTAB 1: PRINT I;" "
    ;P$(I); CHR$ (29);: GOTO 96
    0                                859E
1030 REM ctJCAS SPECIAUXctJ
1040 IF P$(I) = "FIN" THEN IJ =
    I - 1: GOTO 1460                 65C6
1050 IF P$(I) = "***" THEN I = I
    - 2: GOTO 860                    6AF1
1060 REM ctJANALYSE DES PHRASES-
    MISE EN SEGMENTS SIGNIFICAT
    IFSctJ
1070 REM I = NUMERO DE LA PHRASE
1080 REM A$ (I,T) = SEGMENT SIGN
    IFICATIF
1090 REM T = NUMERO DU SEGMENT D
    ANS LA PHRASE
1100 REM NA (I) = NOMBRE DE SEGM
    ENTS DANS LA PHRASE IctJ
1110 A$(I,T) =  LEFT$ (P$(I),1)      A2F3
1120 IF A$(I,T) = "" THEN I = I
    - 1: GOTO 860                    0FE3
1130 K = K + 1:C$ =  MID$ (P$(I)
    ,K,1)                            0EED
1140 IF C$ = " " THEN 1180           0AD6
1150 A$(I,T) = A$(I,T) + C$          FAFD
1160 IF K =  LEN (P$(I)) THEN NA
    (I) = T: GOTO 1180               50CA
1170 GOTO 1130                       3C70
1180 IF A$(I,T) = "POUR" OR A$(I
    ,T) = "DESIRE" OR A$(I,T) =
    "POSSEDE" THEN F = 1: GOTO
    1350                             FBD0
1190 IF T < 2 THEN 1350              8591
1200 IF A$(I,T) = "FAUT" AND A$(
    I,T - 1) = "IL" THEN F = 1:
    GOTO 1290                        C31B
1210 IF  LEN (A$(I,T - 1)) < 3 T
    HEN 1230                         D4E8
1220 IF A$(I,T) = "FAUT" AND  RI
    GHT$ (A$(I,T - 1),2) = "IL"
    THEN X = 2: GOTO 1320            EDC0
1230 IF A$(I,T) = "VENDRE" AND A
    $(I,T - 1) = "VEUT" THEN F
    = 1: GOTO 1290                   945E
1240 IF  LEN (A$(I,T - 1)) < 5 T
    HEN 1260                         F6ED
1250 IF A$(I,T) = "VENDRE" AND
    RIGHT$ (A$(I,T - 1),4) = "V
    EUT" THEN X = 4: GOTO 1320       3607
1260 IF F = 1 THEN F = 0: GOTO 1
    350                              CCAC
1270 IF  RIGHT$ (A$(I,T),1) = "*
    " THEN F = 1: GOTO 1290          D3BD
1280 IF A$(I,T) = "ET" THEN 1350     6D66
1290 A$(I,T - 1) = A$(I,T - 1) +
    " " + A$(I,T):A$(I,T) = ""       5702
1300 IF K =  LEN (P$(I)) THEN NA
    (I) = T - 1: GOTO 1370           EEC5
1310 GOTO 1130                       3C70
1320 A$(I,T) =  RIGHT$ (A$(I,T -
    1),X) + " " + A$(I,T)            EAF9
1330 A$(I,T - 1) =  LEFT$ (A$(I,
    T - 1), LEN (A$(I,T - 1)) -
    X - 1)                           37EF
1340 F = 1: GOTO 1350                1DF5
1350 IF K =  LEN (P$(I)) THEN NA
    (I) = T: GOTO 1370               62CB
1360 T = T + 1: GOTO 1130            4E1B
1370 HOME : PRINT I;" ";             BCAE
1380 FOR T = 1 TO NA(I)              7DC0
1390 IF  RIGHT$ (A$(I,T),1) = "*
    " THEN  PRINT  LEFT$ (A$(I,
    T), LEN (A$(I,T)) - 1); CHR
    $ (44);" ";: GOTO 1410           4970
1400 PRINT A$(I,T);" ";              4313
1410 NEXT T                          A7D6
1420 VTAB 23: PRINT "OK? ¨(O/N)"
```

```
     ;: GET A$                                  0F0D
1430 IF A$ = "O" THEN 860                       74D7
1440 IF A$ <  > "N" THEN 1420                   54CF
1450 P$(I) = "":I = I - 1: GOTO
     860                                        BA3B
1460 REM ctJctJMISE SUR DISKETTE
     ctJctJctJ
1470 GOSUB 420                                  F946
1480 PRINT D$;"OPEN";R$                         8984
1490 PRINT D$;"CLOSE"                           C517
1500 PRINT D$;"DELETE";R$                       6005
1510 PRINT D$;"OPEN";R$                         8984
1520 PRINT D$;"WRITE";R$                        EDDD
1530 PRINT IJ                                   3D4D
1540 FOR I = 1 TO IJ                            2F1F
1550 PRINT NA(I)                                9EE3
1560 FOR T = 1 TO NA(I)                         7DC0
1570 PRINT A$(I,T)                              7F39
1580 NEXT T                                     A7D6
1590 NEXT I                                     9DCB
1600 PRINT D$;"CLOSE"                           C517
1610 REM ctJctJENCHAINEMENTctJct
     JctJ
1620 ZV = 1                                     CDB1
1630 TEXT : HOME                                1C5A
1640 DIM BA$(50),CA$(50)                        996D
1650 IF ZV = 1 THEN 1800                        0CEB
1660 REM ctJctJBASE DE DONNEESct
     JctJctJ
1670 PRINT D$;"CATALOG"                         789C
1680 INPUT "NOM DE LA BASE DE DO
     NNEES DESIREE ? ";R$                       A969
1690 I = 0:X = 0                                99DB
1700 PRINT D$;"OPEN";R$                         8984
1710 PRINT D$;"READ";R$                         D26E
1720 INPUT IJ                                   1117
1730 FOR I = 1 TO IJ                            2F1F
1740 INPUT NA(I)                                AAAD
1750 FOR T = 1 TO NA(I)                         7DC0
1760 INPUT A$(I,T)                              CF03
1770 NEXT T                                     A7D6
1780 NEXT I                                     9DCB
1790 PRINT D$;"CLOSE"                           C517
1800 I$ =  CHR$ (9)                             ECAE
1810 PRINT D$;"PR£1"                            5197
1820 PRINT I$;"80N";                            5A97
1830 IF X = 1 THEN X = 0: GOTO 1
     980                                        6FD9
1840 PRINT "BASE DE DONNEES ";R$
     ;" :": PRINT : PRINT                       0180
1850 FOR J = 1 TO IJ                            4F20
1860 PRINT J;" ";                               D8FE
1870 FOR T = 1 TO NA(J)                         77C1
1880 IF  RIGHT$ (A$(J,T),1) = "*
     " THEN A$(J,T) =  LEFT$ (A$
     (J,T), LEN (A$(J,T)) - 1):
     PRINT A$(J,T); CHR$ (44);"
     ";:A$(J,T) = A$(J,T) + " ":
     GOTO 1920                                  7AB9
1890 A$(J,T) = A$(J,T) + " "                    89FC
1900 PRINT A$(J,T);                             9775
1910 IF  LEFT$ (A$(J,T),3) = "ET
      " THEN A$(J,T) =  MID$ (A$
     (J,T),4, LEN (A$(J,T)) - 1)                309B
1920 NEXT T                                     A7D6
1930 PRINT                                      75BA
1940 NEXT J                                     A3CC
1950 PRINT D$;"PR£3"                            5D99
1960 HOME : VTAB 10: PRINT "TAPE
     Z UNE TOUCHE QUELCONQUE."                  DDD0
1970 GET A$:X = 1: GOTO 1810                    7165
1980 PRINT : PRINT : PRINT                      74A2
1990 PRINT "RAISONNEMENT :": PRI
     NT : PRINT                                 36D3
2000 REM ctJctJDESIREctJctJctJ
2010 W = 1                                      E058
2020 FOR I = W TO IJ                            5245
2030 IF A$(I,0) = "*" THEN 2050                 D5D1
2040 IF A$(I,2) = "DESIRE " THEN
     ZA = 1: GOTO 2080                          DD09
2050 NEXT I                                     9DCB
2060 IF ZA = 0 THEN  PRINT : PRI
     NT "C'EST TOUT !": GOTO 291
     0                                          19B2
2070 ZA = 0: GOTO 2010                          BF43
2080 PRINT :R = I                               2B5F
2090 GOSUB 2790                                 7282
2100 IF A$(I,4) = "POUR " THEN Z
     D = 1: GOTO 2150                           8D96
2110 REM ctJctJPOUR Y IL FAUTctJ
     ctJctJ
2120 FOR J = 1 TO IJ                            4F20
2130 IF A$(J,1) = "POUR " AND A$
     (J,2) = A$(I,3) AND A$(J,3)
     = "IL FAUT " THEN 2230                     27AD
2140 NEXT J                                     A3CC
2150 REM ctJctJPOSSEDE YctJctJct
     J
2160 FOR J = 1 TO IJ                            4F20
2170 IF A$(J,2) = "POSSEDE " AND
     A$(J,3) = A$(I,3) THEN ZC =
     1: GOTO 2380                               48C0
2180 NEXT J                                     A3CC
2190 IF ZC = 0 THEN  PRINT "ET Q
     UE RIEN NE PERMET DE LE SAT
     ISFAIRE,"                                  F82F
2200 PRINT "ALORS ";A$(I,1);"RES
     TE INSATISFAIT ."                          33E5
```

LOGIQUE ÉLÉMENTAIRE (suite)

```
2210 A$(I,0) = "*":ZC = 0:ZB = 0
     :ZD = 0                          161E
2220 GOTO 2050                        6772
2230 R = J                            1D6C
2240 GOSUB 2790                       7282
2250 PRINT "ALORS ";A$(I,1);A$(I
     ,2);A$(J,4);A$(J,1);A$(J,2)      059A
2260 IF NA(J) = 4 THEN 2300           6764
2270 FOR K = 5 TO NA(J)               29BC
2280 PRINT "ET ";A$(I,1);A$(I,2)
     ;A$(J,K);A$(J,1);A$(J,2)         F7C9
2290 NEXT K                           A1CD
2300 A$(I,3) = A$(J,4):A$(I,4) =
     A$(J,1):A$(I,5) = A$(J,2):N
     A(I) = 5                         9984
2310 IF NA(J) = 4 THEN ZB = 0: G
     OTO 2020                         38E4
2320 FOR K = 5 TO NA(J)               29BC
2330 IJ = IJ + 1                      B3EF
2340 A$(IJ,1) = A$(I,1):A$(IJ,2)
     = A$(I,2):A$(IJ,3) = A$(J,K
     ):A$(IJ,4) = A$(J,1):A$(IJ,
     5) = A$(J,2)                     652B
2350 NA(IJ) = 5                       2778
2360 NEXT K                           A1CD
2370 ZB = 0: GOTO 2020                A545
2380 R = J                            1D6C
2390 GOSUB 2790                       7282
2400 REM ctJctJX POSSEDE YctJctJ
     ctJ
2410 IF A$(J,1) = A$(I,1) THEN 2
     570                              19C8
2420 REM ctJctJZ VEUT VENDRE Yct
     JctJctJ
2430 FOR K = 1 TO IJ                  8F21
2440 IF A$(K,1) = A$(J,1) AND A$
     (K,2) = "VEUT VENDRE " AND
     A$(K,3) = A$(J,3) THEN 2480      06AE
2450 NEXT K                           A1CD
2460 PRINT "ET QUE ";A$(J,1);"NE
      VEUT PAS VENDRE ";A$(J,3)       3572
2470 GOTO 2180                        5B76
2480 R = K                            1B6D
2490 GOSUB 2790                       7282
2500 W$ = "A "                        20F0
2510 IF  LEFT$ (A$(J,1),3) = "LE
      " THEN A$(J,1) =  MID$ (A$
     (J,1),4):W$ = "AU "              02CF
2520 PRINT "ALORS ";A$(I,1);"ACH
     ETE ";A$(I,3);W$;A$(J,1)         DC66
2530 PRINT "ET ";A$(I,1);"POSSED
     E ";A$(J,3)                      E39A
2540 A$(J,1) = A$(I,1)                FE89
2550 A$(K,1) = "":A$(K,2) = "":A
     $(K,3) = "":NA(K) = 0            B732
2560 ZB = 0:ZC = 0: GOTO 2020         AD1C
2570 REM ctJctJX EST SATISFAITct
     JctJctJctJ
2580 IF ZD = 1 THEN 2660              52DE
2590 PRINT "ALORS ";                  F6DA
2600 PRINT A$(I,1);"EST SATISFAI
     T ."                             9497
2610 FOR L = 1 TO NA(I)               5EB8
2620 A$(I,L) = ""                     808B
2630 NEXT L                           97CE
2640 NA(I) = 0:ZC = 0:ZD = 0          6FD8
2650 ZB = 0: GOTO 2050                AB48
2660 REM ctJctJD'AUTRES DESIRSct
     JctJctJctJ
2670 FOR L = 1 TO IJ                  8E22
2680 IF A$(L,0) = "*" THEN 2700       0FD6
2690 IF A$(L,1) = A$(I,1) AND A$
     (L,2) = A$(I,2) AND A$(L,3)
     <  > A$(I,3) AND A$(L,4) =
2700 NEXT L                           97CE
2710 PRINT "ALORS ";A$(I,1);"PEU
     T ";A$(I,5)                      47AE
2720 PRINT "ET ";                     43F2
2730 GOTO 2600                        5D73
2740 PRINT "ALORS ";A$(I,1);"POU
     RRAIT ";A$(I,5)                  CFE6
2750 R = L:BA$(R) = "MAIS ":ZD =
     0                                1C28
2760 GOSUB 2820                       777C
2770 A$(I,0) = "*"                    1A99
2780 ZB = 0:ZC = 0:W = L: GOTO 2
     020                              06C9
2790 REM ctJctJS/P RECONSTITUTIO
     N DE PHRASESctJctJctJ
2800 IF ZB = 0 THEN BA$(R) = "CO
     MME ":ZB = 1: GOTO 2820          F084
2810 BA$(R) = "ET QUE "               2A22
2820 FOR T = 1 TO NA(R)               8CC9
2830 BA$(R) = BA$(R) + A$(R,T)        99B4
2840 IF A$(R,2) = "DESIRE " THEN
     2870                             AA98
2850 IF T > 4 AND T < NA(R) THEN
     CA$(R) = CA$(R) +  LEFT$ (A
     $(R,T), LEN (A$(R,T)) - 1)
     + CHR$ (44) + " ": GOTO 288
     0                                BF08
2860 IF T > 4 AND T = NA(R) THEN
```

```
     CA$(R) = CA$(R) + "ET " + A
     $(R,T): GOTO 2880                EA1D
2870 CA$(R) = BA$(R)                  4D65
2880 NEXT T                           A7D6
2890 PRINT CA$(R)                     9F05
2900 RETURN                           63B1
```

```
2910 REM ctJctJFINctJctJ
2920 PRINT : PRINT : PRINT : PRI
     NT : PRINT                       638A
2930 PRINT D$;"PR£3"                  5D99
2940 END                              0180
```

EXEMPLE

Essayez, dans un premier temps, de construire votre propre base de données à partir de ce petit exemple.

```
BASE DE DONNEES SIMPLE :


1  ANDRE DESIRE MANGER
2  POUR MANGER IL FAUT DU PAIN ET DE LA CONFITURE
3  INTERMARCHE POSSEDE DE LA CONFITURE
4  LE BOULANGER POSSEDE DU PAIN
5  INTERMARCHE VEUT VENDRE DE LA CONFITURE
6  LE BOULANGER VEUT VENDRE DU PAIN


RAISONNEMENT :


COMME ANDRE DESIRE MANGER
ET QUE POUR MANGER IL FAUT DU PAIN ET DE LA CONFITURE
ALORS ANDRE DESIRE DU PAIN POUR MANGER
ET ANDRE DESIRE DE LA CONFITURE POUR MANGER

COMME ANDRE DESIRE DU PAIN POUR MANGER
ET QUE LE BOULANGER POSSEDE DU PAIN
ET QUE LE BOULANGER VEUT VENDRE DU PAIN
ALORS ANDRE ACHETE DU PAIN AU BOULANGER
ET ANDRE POSSEDE DU PAIN

COMME ANDRE DESIRE DU PAIN POUR MANGER
ET QUE ANDRE POSSEDE DU PAIN
ALORS ANDRE POURRAIT MANGER
MAIS ANDRE DESIRE DE LA CONFITURE POUR MANGER

COMME ANDRE DESIRE DE LA CONFITURE POUR MANGER
ET QUE INTERMARCHE POSSEDE DE LA CONFITURE
ET QUE INTERMARCHE VEUT VENDRE DE LA CONFITURE
ALORS ANDRE ACHETE DE LA CONFITURE A INTERMARCHE
ET ANDRE POSSEDE DE LA CONFITURE

COMME ANDRE DESIRE DE LA CONFITURE POUR MANGER
ET QUE ANDRE POSSEDE DE LA CONFITURE
ALORS ANDRE PEUT MANGER
ET ANDRE EST SATISFAIT .

C'EST TOUT !
```

Rappelons que ce programme d'initiation a été réalisé dans le cadre des activités du MICROTEL CLUB DE MOURENX.

Vient de paraître

Les Editeurs nous communiquent :

• INITIATION À dBASE III Plus (par R. Cowart traduit de l'américain par J.-P. Cano)

Cet ouvrage permet au néophyte d'apprendre en quelques heures les commandes de base de ***dBASE III Plus***.

Il se présente sous la forme d'un tutorial qui guide l'utilisateur pas à pas dans l'apprentissage des nouvelles fonctions : création d'entrée des données et mise à jour d'une base de données. La description de certaines techniques évoluées permet de sélectionner les informations suivant des critères très spécifiques, d'organiser et de mettre à jour rapidement les fichiers et de créer des rapports de type professionnel.

Cet ouvrage contient :

— recherches élaborées en utilisant les fonctions et les opérateurs de ***dBASE*** ;
— différents niveaux de tris et d'indexages ;
— entrée de commandes à partir du point d'attente ;
— manipulation des nombres, des dates et des champs logiques ;
— linkage de fichiers de données multiples ;
— création de masques de saisies personnalisés pour l'entrée des données.

Tous les exemples de ce livre sont traités sous un aspect professionel : base de données téléphone, étiquettes postales, gestion de stocks, multifichiers, etc....

Un livre broché de 308 pages au format 190X230, 248 F
***SYBEX**, 6-8 impasse du Curé 75018 PARIS.*

• NOUVEAU DICTIONNAIRE dBASE III ET dBASE III Plus (par G. Renner et D. Schmit traduit de l'allemand par J. Bourdeu)

Ouvrage de référence, ce dictionnaire doit essentiellement assister l'utilisateur dans son travail quotidien.

Toutes les commandes et fonctions sont analysées en détail et classées par catégorie d'emploi. Chacune d'entre elles est décrite avec :

— son rôle ;
— la version du logiciel à partir de laquelle elle est disponible ;
— sa syntaxe complète ;
— les conseils d'emploi ainsi que son utilisation avec d'autres commandes ;
— les différences entre ***dBASE III*** et ***dBASE III Plus*** ou entre les versions françaises et américaines ;
— plusieurs exemples détaillés d'utilisation.

L'ouvrage aborde également les commandes, les fonctions réseaux et les utilitaires tels que le générateur de tables et l'éditeur de liens.

Il propose enfin des annexes permettant un accès immédiat aux informations le plus souvent recherchées, telles que la syntaxe exacte d'une commande ou le rôle des touches de contrôle.

Un livre broché de 618 pages au format 190X230, 298 F
***SYBEX**, 6-8 impasse du Curé 75018 PARIS.*

• MS-DOS APPROFONDI (par J. Kamin traduit de l'américain par J.-L. Gréco)

Cet ouvrage s'adresse aux utilisateurs expérimentés de MS-DOS (versions 2.1 à 3.1). Une section traite les commandes les moins familières de ce système d'exploitation.

L'essentiel de l'ouvrage a pour but de familiariser l'utilisateur avec les techniques les plus évoluées lui permettant d'accroître sa productivité : configuration du système, de l'écran et du clavier, élaboration de commandes personnalisées, utilitaires de gestion de disques et de mise au point, gestion d'imprimante, etc...

Il comporte une bibliothèque de sous-programmes prêts à l'emploi et s'impose comme l'outil indispensable de tout utilisateur efficace de MS-DOS.

L'AUTEUR : ***Jonathan Kamin*** *est l'auteur de nombreux ouvrages sur les micro-ordinateurs. Il est également professeur de sociologie et de musique. Il est docteur de l'Université de Princeton.*

Un livre broché de 442 pages au format 190X230, 278 F
***SYBEX**, 6-8 impasse du Curé 75018 PARIS.*

• INTRODUCTION À PASCAL AVEC TURBO PASCAL (par P. Le Beux)

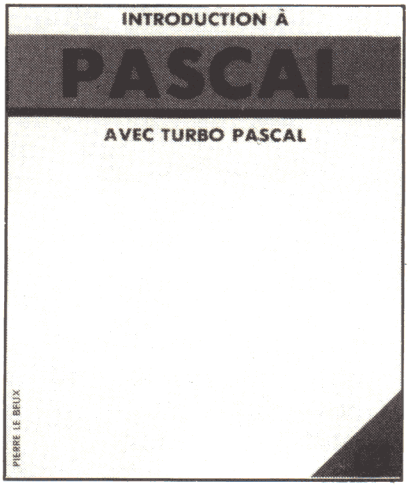
Cet ouvrage s'adresse aux débutants et ne requiert aucune formation préalable aux techniques de l'informatique. Les différents concepts du langage y sont présentés de façon progressive et pédagogique, accompagnés de nombreux exemples.

Pour cette seconde édition, le ***Turbo Pascal*** est le langage de référence. Le livre met en valeur ses qualités intrinsèques de langage simple et structuré. Le traitement graphique y est également abordé.

Il s'agit essentiellement d'un ouvrage de référence.

L'AUTEUR : ***Pierre Le Beux*** *est ingénieur de l'Ecole Centrale et docteur en informatique de l'Université de Californie. Il est l'auteur, chez Sybex, de nombreux best-sellers. Spécialiste des langages de programmation, il est professeur à l'Université de Technologie de Compiègne.*

Un livre broché de 686 pages au format 190X230, 248 F
***SYBEX**, 6-8 impasse du Curé 75018 PARIS.*



UTILITAIRE | DOS | ProDOS

François GALLET

AMPERCOMMAND

Si l'Applesoft est pour la majeure partie d'entre nous un Basic comportant un certain nombre de déficiences, le système dans lequel il navigue permet, heureusement, de pallier la plupart de ses manques. Lorsqu'on se penche sur certains vecteurs de la page 3 ($Ø3ØØ à $Ø3FF), grande est notre satisfaction en constatant que l'un d'entre eux, le fameux Ampersand, permet d'ajouter des commandes nouvelles à ce bon vieil Applesoft.

AMPERCOMMAND fonctionne aussi bien en DOS 3.3 qu'en PRODOS et met à votre disposition, dans sa version de base, la possibilité d'afficher des fenêtres de n'importe quelle taille (dans les limites de l'écran texte, bien sûr !), à n'importe quel emplacement de l'écran. Le nombre des fenêtres affichables simultanément est illimité et ne dépend que de votre humeur.

SYNTAXE

La syntaxe est très simple : & FENETRE BG,BD,LH,LB

BG = Bord gauche de la fenêtre que l'on désire afficher.
BD = Bord droit.
LH = Ligne du haut.
LB = Ligne du bas.

Cette commande est active en mode direct ou en mode programme.

AMPERCOMMAND apporte également un support à tous ceux qui, pratiquant l'Assembleur, désirent ajouter des instructions supplémentaires à l'Applesoft. On peut ainsi placer jusqu'à 256 commandes nouvelles, comme par exemple, & PRINT USING, & DHGR (Double Haute Résolution Graphique), etc. La longueur de la syntaxe importe peu (255 caractères maxi), les espaces, s'ils n'ont pas été définis comme partie intégrante de la commande, sont ignorés lors de l'exécution.

COMMENT AJOUTER UNE COMMANDE ?

Pour cela, servez-vous du source d'*AMPERCOMMAND* et procédez ainsi :

1° Reportez-vous à la *Table des commandes* (ligne 281). Insérez le texte de votre commande, si c'est la première que vous ajoutez, à la suite du pointeur de fin de commande (octet $ØØ) de FENETRE. Les octets composant votre commande devront être codés en ASCII bas (bit 7 à Ø) et se terminer par un $ØØ.

Exemple : Vous ajoutez la commande & EFFACE. Ecrivez ASC 'EFFACE' suivi de HEX ØØ. Le codage hexadécimal de EFFACE devra être celui-ci : 45 46 46 41 43 45 suivi de ØØ.

2° Il faut maintenant indiquer à *AMPERCOMMAND* que vous avez ajouté une commande et préciser où trouver le programme de traitement. Reportez-vous à la ligne 118 du source. En arrivant ici, *AMPERCOMMAND* confirme qu'il a bien trouvé la correspondance entre ce qui est dans le buffer ($Ø2ØØ) et quelque chose dans la table, le registre X contenant le numéro d'ordre de la commande trouvée. Reprenons l'exemple de & EFFACE :

```
NOERROR   CPX  £$ØØ     ; Commande & FENETRE.
          BNE  CDE2     ; Si X <> Ø, on passe à la
                          commande n°2.
```

```
        JMP  FENETRE ; Vers traitement de &
                       FENETRE.
CDE2    CPX  £$01    ; Commande & EFFACE.
        BNE  CDE3    ; Si X <> 1, on passe à la
                       commande n°3.
        JMP  EFFACE  ; Vers traitement de &
                       EFFACE.
*
CDE3    NOP          ; Autres commandes...
```

Votre nouvelle commande est opérationnelle... si le traitement existe, bien sûr !

3° Les traitements d'erreurs se trouvent à la ligne 108. Si vous désirez que l'Applesoft vous renvoie un de ses propres messages, mettez dans le registre X le code de l'erreur et retournez au système par JMP ERROR.

4° Faites en sorte que votre traitement de commande se termine par l'instruction "JMP BASIC" afin de pouvoir rendre la main au système dès l'exécution achevée de votre nouvelle commande.

La routine a été implantée en $8000, mais peut être assemblée n'importe où en mémoire, selon la masse de codes que vous ne manquerez pas de rajouter au fur et à mesure de votre développement. Notez qu'*AMPERCOMMAND* repositionne la valeur de HIMEM pour se protéger et qu'un RESET malencontreux ne le perturbe pas. Pour rendre actif *AMPERCOMMAND*, faites "CALL 32768" en Basic, ou "$8000G" sous Moniteur.

En ce qui concerne la commande *& FENETRE*, une courte présentation valant mieux qu'un long discours, tapez le programme Basic qui suit et regardez...

François GALLET.

AMPERCOMMAND a été assemblé par ProCODE, mais un autre assembleur fera l'affaire. Si vous ne disposez pas de cet outil, contentez-vous de taper les codes de la page 44.

```
 0  **********************
 1  *                    *
 2  *    AMPERCOMMAND    *
 3  *                    *
 4  **********************
 5  *
 6  *
 7  *    Auteur : Francois GALLET
 8  *    ------
 9  *
10  AMPER    EQU  $03F5      ;Adresse du vecteur Ampersand
11  BASCALC  EQU  $FBC1      ;Calcule l'adresse de base dans l'écran texte
12                           ;d'une ligne dont le numéro est dans l'accu.
13  BASIC    EQU  $D995      ;Retour à l'interpréteur pour la prochaine ins-
14                           ;-truction
15  BTM      EQU  $23        ;Borne basse de la fenêtre de texte
16  CH       EQU  $24        ;Stockage du HTAB
17  CHRGET   EQU  $00B1      ;Incrémente d'abord TXTPTR ($B8,$B9) . Retourne
18                           ;dans le registre "A" la valeur de l'octet dont la
19                           ;position dans le buffer ($0200) est donnée par
20                           ;TXTPTR
21  HOME     EQU  $FC58      ;Efface l'écran
22  CV       EQU  $25        ;Stockage du VTAB
23  ERROR    EQU  $D412      ;Traitement de l'erreur dont le code est dans X
24  GETBYTC  EQU  $E6F5      ;Convertit une expression numérique (inférieure à
25                           ;256) en entier . Au retour , X contient la valeur
26  HIMEM    EQU  $0073      ;Adresse la plus élevée de la RAM
27  PAGE1    EQU  $C054      ;Commutateur 'Ecran texte mémoire principale'
28  PAGE1X   EQU  $C055      ;Commutateur 'Ecran texte mémoire auxiliaire'
29  VTABZ    EQU  $FC24      ;Positionne le curseur en fonction de CV
30  WDTH     EQU  $21        ;Largeur de la fenêtre de texte
31  WNDLFT   EQU  $20        ;Marge gauche de la fenêtre de texte
```

```
32  WNDTOP   EQU  $22          ;_____ haute  _____________________
33  *
34           ORG  $8000
35  *
36  *   Initialisation du vecteur Ampersand avec l'adresse de début d'AMPERCDE
37  *
38  DEBUT    LDA  £<AMPERCDE
39           STA  AMPER+1
40           LDA  £>AMPERCDE
41           STA  AMPER+2
42  *
43  *   Protection du programme par revectorisation de HIMEM
44  *
45           LDA  £<DEBUT
46           STA  HIMEM
47           LDA  £>DEBUT
48           STA  HIMEM+1
49  *
50           RTS
51  *
52  *   Initialisation de l'adresse de départ de la table des commandes
53  *
54  AMPERCDE LDA  £<COMMAND
55           STA  $19
56           LDA  £>COMMAND
57           STA  $1A
58  *
59  *   La syntaxe de la commande est-elle correcte ?
60  *
61           LDX  £$00         ;Compteur de commande (256 maxi)
62           DEC  $B8          ;A cause du CHRGET qui suit
63  *
64           LDA  $B8
65           STA  $1B
66           LDA  $B9
67           STA  $1C
68  *
69           LDY  £$FF
70  LOOP1    INY
71  LOOP4    LDA  ($19),Y      ;Test de fin de commande (pointeur nul)
72           BEQ  NOERROR      ;Si fin , on poursuit !
73           JSR  CHRGET       ;On prend un caractère contenu dans le buffer...
74           CMP  ($19),Y      ;...et on compare à la syntaxe de la commande
75           BEQ  LOOP1        ;Si les deux caractères sont équivalents , on
76                             ;boucle , sinon ...
77  *
78  *    Préparation de la réinitialisation et test d'erreur
79  *
80  LOOP2    INY
81           LDA  ($19),Y      ;Fin de commande ?
82           BNE  LOOP2        ;On boucle tant que l'on n'y est pas...
83           INY
84           LDA  ($19),Y      ;Fin de table ?
```

```
85            BEQ  SYNTAX      ;Si oui , il y a une erreur de syntaxe !
86  *
87  *
88  *    Réinitialisation de l'adresse de début de la table des commandes pour
89  *    la prise en compte d'une nouvelle commande .
90  *
91  LOOP3     INC  $19
92            BNE  NOZERO
93            INC  $1A
94  NOZERO    DEY
95            BNE  LOOP3
96  *
97  *    On remet le pointeur de CHRGET à son ancienne valeur
98  *
99            LDA  $1B
100           STA  $B8
101           LDA  $1C
102           STA  $B9
103 *
104           INX              ;Commande suivante
105 *
106           JMP  LOOP4
107 *
108 *   Traitement de l'erreur rencontrée (d'autres types sont possibles selon
109 *   la commande)
110 *
111 SYNTAX    LDX  £$10        ;"SYNTAX ERROR"
112           JMP  ERROR       ;...
113 *
114 ILLEGAL   LDX  £$35        ;"ILLEGAL QUANTITY ERROR"
115           JMP  ERROR       ;...
116 *
117 *
118 *   Quelle est la commande ?
119 *
120 NOERROR   CPX  £$00
121           BNE  CDE2
122           JMP  FENETRE
123 CDE2      NOP              ;Autres commandes (256 en tout)
124 *
125 *   Commande : & FENETRE CG,CD,LH,LB
126 *   -------------------------------
127 *    Evaluation de l'expression qui suit la commande "& FENETRE" .
128 *
129 FENETRE  JSR  GETBYTC
130           CPX  £79
131           BCS  ILLEGAL
132           STX  COLMIN
133           JSR  GETBYTC
134           CPX  £80
135           BCS  ILLEGAL
136           STX  COLMAX
137           JSR  GETBYTC
```

```
138             CPX  £23
139             BCS  ILLEGAL
140             STX  LINMIN
141             JSR  GETBYTC
142             CPX  £25
143             BCS  ILLEGAL
144             STX  LINMAX
145   *
146   *    On évite que la borne gauche et la borne haute + 1 de la fenêtre ne
147   *    soit respectivement supérieure ou égale à la borne droite et la borne
148   *    basse .
149   *
150             LDA  COLMIN
151             CMP  COLMAX
152             BCS  ILLEGAL
153             LDA  LINMIN
154             INA
155             CMP  LINMAX
156             BCS  ILLEGAL
157   *
158             JSR  CADRE        ;ENFIN !!!
159   *
160             JMP  BASIC        ;Prochaine commande Basic
161   *
162   *
163   *    Sous programmes : CADRE  (Tracé d'un cadre avec effacement de la
164   *    ----------------------    fenêtre)
165   *
166   *    Sauvegarde de la position courante du curseur
167   *
168   CADRE     LDA  CV
169             PHA
170             LDA  CH
171             PHA
172   *
173   *    Effacement de la fenêtre
174   *
175             LDA  COLMIN       ;Nouvelle marge gauche
176             STA  WNDLFT       ;...
177             LDA  COLMAX       ;Nouvelle largeur
178             SEC               ;...
179             SBC  WNDLFT       ;...COLMAX - WNDLFT = LARGEUR
180             STA  WDTH         ;...
181             LDA  LINMIN       ;Nouvelle marge haute
182             STA  WNDTOP       ;...
183             LDA  LINMAX       ;Nouvelle marge basse
184             STA  BTM          ;...
185             JSR  HOME         ;Enfin !
186   *
187             STZ  WNDLFT       ;Marge gauche standard
188             STZ  WNDTOP       ;Marge haute standard
189             LDA  £$50         ;Largeur standard
190             STA  WDTH         ;...de la fenêtre
```

```
191             LDA  £$18       ;Marge basse standard
192             STA  BTM
193   *
194   *   Tracé du cadre de la fenètre
195   *
196             LDA  £$4C       ;Code de l'icone pour trait horizontal supérieur
197             STA  ICONE      ;...
198             LDA  LINMIN     ;Ligne du titre de la fenêtre à tracer
199             STA  LIGNE      ;...
200             JSR  HORIZ      ; --> initialisation et tracé
201             LDA  £$DF       ;Code de l'icone pour trait horizontal inférieur
202             STA  ICONE      ;...
203             LDA  LINMAX     ;Ligne inférieure
204             DEA
205             STA  LIGNE      ;...
206             JSR  HORIZ      ; --> initialisation et tracé
207   *
208             LDA  £$5A       ;Code de l'icone pour verticale gauche
209             STA  ICONE      ;...
210             LDA  COLMIN     ;Colonne gauche
211             STA  COLONNE    ;...
212             JSR  VERTIC     ; --> initialisation et tracé
213             LDA  £$5F       ;Code de l'icone pour verticale droite
214             STA  ICONE      ;...
215             LDA  COLMAX     ;Colonne droite
216             STA  COLONNE    ;...
217             JSR  VERTIC     ; --> initialisation et tracé
218   *
219   *   Restauration de la position du curseur
220   *
221             PLA
222             STA  CH
223             PLA
224             STA  CV
225             JSR  VTABZ      ;Placement du curseur à l'écran
226   *
227             RTS             ;Retour d'appel
228   *
229   *
230   *   Sous programme : HORIZ (tracé des horizontales du cadre)
231   *   ------------------------------------------------------
232   *
233   HORIZ     LDA  COLMIN     ;Colonne de gauche...
234             INA             ;...+1...
235   LOOPH     PHA             ;On sauve pour traitement suivant
236             JSR  COM.LOG    ;Mémoire principale ou auxiliaire ?
237             LDA  LIGNE      ;N° de ligne...
238             JSR  BASCALC    ;...et son adresse à l'écran
239             LDY  COL40      ;N° de colonne en 40 colonnes
240             LDA  ICONE      ;Code du caractère à afficher
241             STA  (£28),Y    ; --> Affichage
242             PLA             ;On restaure la colonne...
243             INA             ;...+1
```

```
244             CMP  COLMAX      ;Est-ce que l'on a atteint la borne à droite ?
245             BMI  LOOPH       ;Si ce n'est pas le cas , on boucle
246             BIT  PAGE1       ;...au cas ou !
247             RTS              ;Sinon , retour d'appel
248   *
249   *   Sous programme : VERTIC (tracé des verticales du cadre)
250   *   ----------------------------------------------------
251   *
252   VERTIC    LDA  COLONNE     ;Colonne du tracé...
253             JSR  COM.LOG     ;...en PAGE 1 ou 1X ?
254             LDA  LINMIN      ;Ligne de départ...
255   LOOPV     PHA              ;On sauve
256             JSR  BASCALC     ;...et son adresse à l'écran
257             LDY  COL40       ;N° de colonne en 40 colonnes
258             LDA  ICONE       ;Code de l'icone à afficher
259             STA  ($28),Y     ; --> Affichage
260             PLA              ;On restaure la ligne de tracé
261             INA              ;...+1
262             CMP  LINMAX      ;Dernière ligne ?
263             BMI  LOOPV       ;Non alors on recommence
264             BIT  PAGE1       ;...au cas ou !
265             RTS              ;Si oui , alors retour d'appel
266   *
267   *   Sous programme : COM.LOG (commute en PAGE 1 si la colonne est impaire
268   *   ------------------------  ou en PAGE 1X si elle est paire)
269   *
270   COM.LOG   ROR              ;Un décalage à droite pour tester le bit de
271                              ;poids faible
272             BCS  MEM         ;S'il vaut 1 , le nombre est impair (Lapalisse)
273             BIT  PAGE1X      ;S'il vaut 0 , le nombre est pair --> Page 1X...
274             BRA  CONT4       ;...et on saute la commutation en Page 1
275   MEM       BIT  PAGE1       ;Commutation en Page 1 si nombre impair
276   CONT4     ROL              ;On restitue le bit de droite...
277             LSR              ;...on décale à droite pour diviser par 2
278             STA  COL40       ;et on stocke le résultat (en 40 colonnes)
279             RTS              ;Retour d'appel
280   *
281   *   Table des commandes (la longueur du texte de la commande est
282   *   -------------------  quelconque)
283   *


284   COMMAND   ASC  'FENETRE'   ;Syntaxe de la commande & FENETRE (bit 7 à 0)
285             HEX  00          ;Pointeur de fin de commande
286   *
287             HEX  00          ;Pointeur de fin de la table des commandes
288   *
289   *
290   *   Stockage de paramètres
291   *   ----------------------
292   *
293   COL40     HEX  00          ;Mémorise l'abcisse comprise entre 0 et 39
294   COLMAX    HEX  00          ;Colonne droite de la fenêtre
```

```
295  COLMIN   HEX  00          ;Colonne gauche de la fenêtre
296  COLONNE  HEX  00          ;Utilisé par VERTIC
297  ICONE    HEX  00          ;Stockage du code de l'icone à afficher
298  LIGNE    HEX  00          ;Utilisé par HORIZ
299  LINMAX   HEX  00          ;Ligne haute de la fenêtre
300  LINMIN   HEX  00          ;Ligne basse de la fenêtre
```

AMPERCDE.C,A$8000,L$0177

```
8000: A9 13 8D F6 03 A9 80 8D F7 03 A9 00 85 73 A9 80   1FBC
8010: 85 74 60 A9 66 85 19 A9 81 85 1A A2 00 C6 B8 A5   C194
8020: B8 85 1B A5 B9 85 1C A0 FF C8 B1 19 F0 30 20 B1   B679
8030: 00 D1 19 F0 F4 C8 B1 19 D0 FB C8 B1 19 F0 15 E6   6FA8
8040: 19 D0 02 E6 1A 88 D0 F7 A5 1B 85 B8 A5 1C 85 B9   8A36
8050: E8 4C 2A 80 A2 10 4C 12 D4 A2 35 4C 12 D4 E0 00   E7AB
8060: D0 03 4C 66 80 EA 20 F5 E6 E0 4F B0 EC 8E 71 81   FA35
8070: 20 F5 E6 E0 50 B0 E2 8E 70 81 20 F5 E6 E0 17 B0   56DE
8080: D8 8E 76 81 20 F5 E6 E0 19 B0 CE 8E 75 81 AD 71   BF71
8090: 81 CD 70 81 B0 C3 AD 76 81 1A CD 75 81 B0 BA 20   74BD
80A0: A5 80 4C 95 D9 A5 25 48 A5 24 48 AD 71 81 85 20   8E46
80B0: AD 70 81 38 E5 20 85 21 AD 76 81 85 22 AD 75 81   AC6F
80C0: 85 23 20 58 FC 64 20 64 22 A9 50 85 21 A9 18 85   960B
80D0: 23 A9 4C 8D 73 81 AD 76 81 8D 74 81 20 14 81 A9   C71D
80E0: DF 8D 73 81 AD 75 81 3A 8D 74 81 20 14 81 A9 5A   CE77
80F0: 8D 73 81 AD 71 81 8D 72 81 20 35 81 A9 5F 8D 73   8C7E
8100: 81 AD 70 81 8D 72 81 20 35 81 68 85 24 68 85 25   0998
8110: 20 24 FC 60 AD 71 81 1A 48 20 55 81 AD 74 81 20   7759
8120: C1 FB AC 6F 81 AD 73 81 91 28 68 1A CD 70 81 30   BC22
8130: E7 2C 54 C0 60 AD 72 81 20 55 81 AD 76 81 48 20   6B29
8140: C1 FB AC 6F 81 AD 73 81 91 28 68 1A CD 75 81 30   A227
8150: ED 2C 54 C0 60 6A B0 05 2C 55 C0 80 03 2C 54 C0   48B0
8160: 2A 4A 8D 6F 81 60 46 45 4E 45 54 52 45 00 00 00   2A5A
8170: 00 00 00 00 00 00 00                              0000
```

Si vous ne possédez pas ce précieux outil qu'est un bon assembleur, contentez-vous de recopier ces codes, puis sauvez votre routine sur disquette.

N'oubliez pas que la vérification est franchement facilitée en utilisant **SIGNATURE.**

CONSEILS AUX DÉBUTANTS :

Rappelons que, pour taper un programme en langage machine (sans assembleur), il suffit d'accéder au Moniteur par un **CALL - 151**, puis de taper textuellement chacune des lignes ci-dessus. Lorsque cette tâche — ô combien épuisante — est terminée, sortir du Moniteur par CTRL-C (touche CTRL + C), puis taper le **BSAVE NOM,A$7000,L$01FC** indiqué en tête de programme.

PRÉCAUTION : Avant de rentrer les codes d'un programme, s'assurer que l'on a bien accès au DOS ou à ProDOS en essayant d'obtenir le catalogue de la disquette. Eventuellement (cela n'engage à rien !) essayer de sauver la routine avant même de l'avoir tapée. Rien n'est plus désagréable que de se retrouver devant un écran rempli de codes, après vingt minutes de frappe laborieuse... sans le DOS !

Attention ! Ne tapez surtout pas les valeurs indiquées en couleur, à la fin de chaque ligne : elles sont destinées aux utilisateurs de la disquette SIGNATURE et permettent de localiser très rapidement les éventuelles erreurs de saisie (lire, à ce sujet le bas de la page précédente). ■

AMPERCDE.DEMO

```
 10 REM *** DEMO AMPERCOMMAND ***
 20 :                                                    003A
 30 D$ =  CHR$ (4)                                       B3A4
 40 PRINT D$;"BLOAD AMPERCDE.C"                          8BD5
 50 CALL 32768: REM --- Initialisation ---                7196
 60 HOME : LIST 10 - 180                                 2050
 70 & FENETRE0,76,3,7                                    1943
 80 VTAB 5: HTAB 3: PRINT "Ce programme ,
    écrit en Assembleur , a pour ";                      70B1
 90 PRINT "fonction de vous permettre de"                B4F5
100 VTAB 6: HTAB 3: PRINT "tracer des fenè
    tres..."                                             E10E
110 GOSUB 1000                                           5971
120 & FENETRE30,71,6,13                                  98A1
130 VTAB 8: HTAB 33: PRINT "...que vous po
    urrez définir simplement"                            CCF6
140 VTAB 9: HTAB 33: PRINT "en tapant une
    nouvelle commande dont"                              0C0F
150 VTAB 10: HTAB 33: PRINT "la syntaxe es
    t :"                                                 67B0
160 VTAB 12: HTAB 33: PRINT "--> & FENETRE
     BG,BD,LH,LB"                                        EB4F
165 GOSUB 1000                                           5971
170 & FENETRE5,26,9,17                                   E27A
180 VTAB 11: HTAB 8: PRINT "...ou :"                     92CC
190 VTAB 13: HTAB 8: PRINT "BG = Bord gauc
    he"                                                  0640
200 VTAB 14: HTAB 8: PRINT "BD = Bord droi
    t"                                                   6F13
210 VTAB 15: HTAB 8: PRINT "LH = Ligne du
    haut"                                                B9D3
220 VTAB 16: HTAB 8: PRINT "LB = Ligne du
    bas"                                                 DD72
230 GOSUB 1000                                           5971
240 & FENETRE26,77,13,23                                 E2DB
250 VTAB 15: HTAB 29: PRINT "La commande p
    eut s'employer en mode direct ou en"                 E11B
260 VTAB 16: HTAB 29: PRINT "mode programm
    e . Elle peut être suivie par une"                   03E1
270 VTAB 17: HTAB 29: PRINT "autre instruc
    tion Basic ."                                        CE50
280 VTAB 19: HTAB 29: PRINT "ATTENTION :"l
    a partie d'écran recouverte par la"                  DB1F
290 VTAB 20: HTAB 29: PRINT "fenêtre sera
    définitivement perdue ."                             B1A7
300 VTAB 21: HTAB 29: PRINT "Si vous désir
    ez le retrouver , utilisez une"                      EB79
310 VTAB 22: HTAB 29: PRINT "sauvegarde d'
    écran ."                                             36D8
320 GOSUB 1000: GOSUB 900                                40F4
330 & FENETRE7,54,3,10                                   6670
340 VTAB 5: HTAB 10: PRINT "Il est importa
    nt de noter que cet utilitaire"                      DB33
350 VTAB 6: HTAB 10: PRINT "a été écrit po
    ur ceux d'entre vous possédant"                      8DF1
360 VTAB 7: HTAB 10: PRINT "un IIe+ , un I
    Ic ou un II GS ."                                    604C
370 VTAB 8: HTAB 10: PRINT "AMPERCOMMAND f
    onctionne en 80 colonnes , ce"                       FC7A
380 VTAB 9: HTAB 10: PRINT "qui est tous d
    e même plus agréable ."                              1F0A
390 GOSUB 1000                                           5971
400 & FENETRE12,68,10,16                                 69D5
410 VTAB 12: HTAB 15: PRINT "Si vous conna
    issez l'Assembleur et que vous possédi
    ez"                                                  055C
420 VTAB 13: HTAB 15: PRINT "de quoi créer
     un source (ProCODE) , AMPERCOMMAND vo
    us"                                                  D2A0
430 VTAB 14: HTAB 15: PRINT "permet de raj
    outer de nouvelles commandes , qui"                  FF0F
440 VTAB 15: HTAB 15: PRINT "associées à l
    'Ampersand , enrichiront l'Applesoft .
    "                                                    7819
450 GOSUB 1000                                           5971
460 & FENETRE2,42,17,20                                  E79E
470 VTAB 19: HTAB 5: PRINT "Et maintenant
    , un peu de mouvement !"                             18D4
480 GOSUB 900                                            2349
490 HOME                                                 2F97
500 FOR K = 1 TO 10                                      ECEF
510 BG = 0:BD = 79:LH = 0:LB = 23: FOR N =
    1 TO 10                                              8280
520 BG = BG + 1:BD = BD - 1:LH = LH + 1:LB
    = LB - 1: & FENETREBG,BD,LH,LB: NEXT                 B199
530 FOR TEMPO = 0 TO 500: NEXT : NEXT K                  D11F
540 GOSUB 900                                            2349
660 & FENETRE8,71,17,22                                  91A8
670 VTAB 19: HTAB 11: PRINT "Bonne program
    mation , et à bientot ..."                           672C
680 VTAB 21: HTAB 50: PRINT "Francois GALL
    ET"                                                  E7A0
690 GOSUB 1000: GOSUB 900: HOME                          EFC5
700 END                                                  0180
900 & FENETRE25,52,10,14                                 D2D0
910 VTAB 12: HTAB 29: INVERSE : PRINT "APP
    UYEZ SUR UNE TOUCHE": NORMAL                         995F
920 VTAB 13: HTAB 29: INVERSE : PRINT "POU
    R CONTINUER...";: NORMAL                             CA0A
930 GET T$: PRINT : HOME : RETURN                        CDE6
1000 FOR TEMPO = 0 TO 5000: NEXT TEMPO                   13CD
1010 RETURN                                              63B1
```

Jean-Paul VERPEAUX

INTERFACE MIDI

pour Apple *IIe, IIc, II+* et *GS*

L'INTERFACE MIDI

MIDI est l'Acronyme de Musical Instrument Digital Interface. Cette interface permet d'interconnecter des synthétiseurs, boîtes à rythmes, et autres instruments ayant adopté les normes MIDI. Certains ordinateurs à vocation musicale ont déjà leur interface MIDI intégrée ; comme ce n'est malheureusement pas le cas des divers ordinateurs Apple, nous avons décidé d'y remédier en vous proposant les plans d'une interface MIDI simple et bon marché.

Chaque interface MIDI comprend un émetteur (prise OUT) et un récepteur (prise IN) et une ou plusieurs prises THRU chargées de recopier le signal d'entrée pour l'adresser à d'autres instruments câblés en chaîne. Les normes MIDI prévoient entre autres que l'étage d'entrée soit protégé par un photocoupleur.

HARDWARE.

L'interface MIDI est en fait une carte série un peu particulière dans le sens qu'elle travaille à une vitesse inhabituellement élevée : 31,25 KBauds.

Le cœur du montage est constitué principalement d'un convertisseur série/parallèle et vice-versa appelé UART (Universal Asynchronous Receiver Transmitter) ou encore ACIA.

Les normes MIDI imposent de travailler avec une boucle de courant de l'ordre de 5mA. Les niveaux logiques Ø sont représentés par ce courant circulant dans les câbles de liaison. Inversement, les niveaux logiques 1 correspondent à des absences de courant.

Un message MIDI comprend 8 bits. Ces bits sont transmis avec un bit de start et sont suivis d'un bit de stop, soit un total de 1Ø bits par information à transmettre. Chaque bit transmis dure 32 microsecondes ; il faudra donc 32Ø microsecondes pour transmettre un octet.

Le montage que nous vous proposons peut être réalisé de plusieurs façons :

— Soit sur une carte d'étude, enfichable dans un des slots de l'Apple.

— Soit sur un circuit imprimé ou une carte d'étude, extérieure à l'Apple et reliée à ce dernier par un connecteur 5Ø contacts et un câble en nappe de courte longueur. Cette dernière solution est plus pratique et plus esthétique. ■

UTILISATION DE L'INTERFACE

L'interface MIDI comporte 4 registres :

— Registre d'émission,
— Registre de réception,
— Registre de contrôle,
— Registre de statut.

Ces 4 registres seront vus par l'Apple comme deux adresses mémoire successives. Deux adresses suffisent, car chacun de ces registres ne peut qu'être lu ou écrit (jamais les deux), ainsi :

- Contrôle et Statut se partagent l'adresse CØA8.
- Emission et Réception se partagent l'adresse CØA9.

Ces deux valeurs arbitraires ont été choisies pour rendre notre interface compatible avec les logiciels MIDI du commerce qui fonctionnent avec des interfaces MIDI Yamaha ou Passport-Design. Cette compatibilité impose également que l'interface MIDI soit placée dans le slot 2.

Pour utiliser l'interface, il faut tout d'abord la programmer conformément aux normes MIDI (vitesse de transmission, nombre de bits de stop, etc.). Pour cela, on doit écrire successivement deux valeurs dans le registre de contrôle :

```
LDA § Ø3      Effectue la remise à zéro de l'ACIA,
STA CØA8      donc de l'interface.
```

Ensuite :

```
LDA § 15      ; 15 Hexa, soit ØØØ1 Ø1Ø1 en binaire.
STA CØA8
```

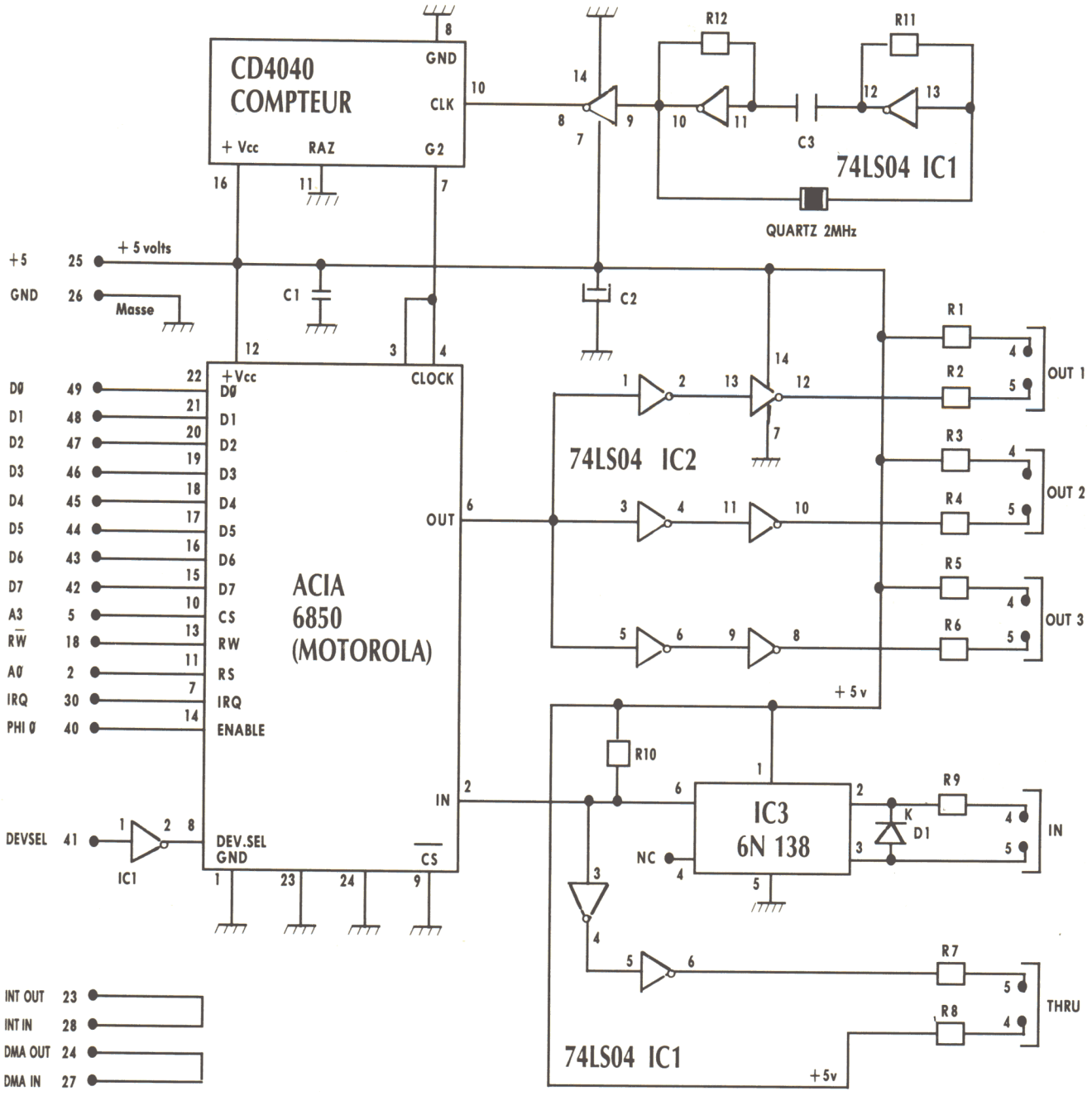
Ou encore :

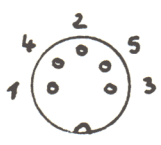
```
LDA § 95      ; 95 Hexa, soit 1ØØ1 Ø1Ø1 en binaire.
STA CØA8
```

(suite page 48)

INTERFACE MIDI pour Apple

J.-P. VERPEAUX (1987)





Prises DIN IN/OUT/THRU

Le premier cas correspond à la programmation standard de l'interface, utilisée surtout pour émettre des codes. Dans le deuxième cas, l'ACIA est programmé pour créer des demandes d'interruptions, sur la ligne IRQ, lorsque son registre de réception est plein. C'est très pratique, voire même indispensable lorsque l'Apple doit recevoir les codes MIDI émis par un instrument périphérique.

NOTA : La programmation peut aussi se faire en Basic, en pokant les valeurs décimales correspondant à § Ø3, § 15 ou § 95, à l'adresse 49320.

Ensuite, il vous reste à envoyer vos messages, ou lire ceux qui arrivent, en écrivant ou en lisant à l'adresse CØA9. L'émission de codes MIDI peut aussi se faire en Basic, en pokant à l'adresse 49321 les valeurs désirées.

Par contre, le Basic ne permet pas de lire les codes qui arrivent, ces derniers se présentant dans l'ACIA à une cadence trop élevée. En réception, le langage machine est indispensable.

EMISSION DE CODES

En théorie, il suffit de faire :

```
LDA § xx  ; xx représente le code à envoyer.
STA CØA9
```

En pratique, avant d'émettre un code, il faut s'assurer que la transmission du code précédant est entièrement achevée (souvenez-vous qu'un code met 320 microsecondes pour être entièrement transmis).

Cela impose une petite routine de test ; en voici un exemple simple :

```
LOOP    LDA  CØA8   ; On lit le registre de statut.
        AND  § Ø2   ; Test du bit 1 de ce registre
                    (registre émission encore
                    plein).
        BEQ  LOOP   ; On boucle tant que le
                    registre d'émission n'est
                    pas vide.
READY   LDA  § xx   ; xx = valeur à émettre.
        STA  CØA9   ; Emission du code XX.
        RTS
```

RÉCEPTION DES CODES

Les codes reçus sont réceptionnés par le registre de réception qui peut être lu à l'adresse CØA9 (même adresse que le registre d'émission). On peut savoir si un code est entièrement arrivé en testant le bit 1 du registre de statut (en CØA8) ; d'où la routine de réception suivante :

```
LOOP    LDA  CØA8   ; Lecture du registre de sta-
                    tut.
        AND  § Ø1   ; Test du bit Ø (registre
                    réception plein ?).
        BEQ  LOOP   ; Message pas encore
                    arrivé : on boucle.
LIT     LDA  CØA9   ; On lit le code réceptionné.
        STA  AD     ; On le range à l'adresse
                    AD par exemple.
        RTS
```

En réception, on peut réceptionner les codes soit en scrutant en permanence le registre de statut (son bit 1), ou mieux encore, en détournant les interruptions IRQ pour ranger les codes chaque fois qu'ils se présentent. Cette dernière solution, plus compliquée mais plus efficace sera vue dans un autre article. Nous vous indiquons toutefois, si vous voulez l'expérimenter, qu'il faut pour cela programmer l'ACIA avec la valeur § 95 (Hexa) dans le registre de contrôle (CØA8). Ensuite, au lieu de tester le bit Ø du registre de statut, on testera le bit 7 pour s'assurer que l'IRQ demandée est bien due à notre interface MIDI qui vient de réceptionner un code.

BIBLIOGRAPHIE : ***"Midi"*** et ***"Technique des synthétiseurs"***, par J.-P. VERPEAUX, Editions Musicom, chez tous les bons marchands de musique.

REGISTRE DE STATUT

Ce registre se partage l'adresse CØA8 avec le registre de contrôle de l'ACIA ; il ne peut qu'être lu, tandis que le registre de contrôle ne peut qu'être écrit.

Chacun des bits de ce registre sert de drapeau et est mis à 1 dans des circonstances bien précises.

BIT Ø Indique que le registre de réception est plein.

BIT 1 Indique que le registre d'émission est vide.

BIT 4 Indique une erreur de cadrage (par exemple lorsque l'horloge des deux interfaces qui dialoguent n'est pas identique).

BIT 5 Surcharge du récepteur : si un mot reçu n'est pas lu assez rapidement, l'apparition d'un nouveau message entraîne une surcharge du récepteur.

BIT 7 Interruption demandée par l'ACIA.

Les autres bits correspondent à des fonctions non utilisées par le système MIDI ; l'ACIA 6850 pouvant avoir d'autres utilisations (par exemple : modem).

REGISTRE DE CONTRÔLE

Ce registre sert à programmer l'ACIA, conformément aux normes MIDI. En écrivant la valeur binaire 11 dans les bits 0 et 1, on remet à zéro l'ACIA ; d'où la valeur 03 utilisée pour la RAZ de l'interface.

Les bits 0 et 1 servent ensuite à définir le facteur de division interne de l'ACIA. Pour obtenir une fréquence de 32,125 KHz, l'oscillateur à quartz de 2MHz est d'abord divisé par 4 par le CD 4040 ; ce qui donne du 500 KHz, puis cette valeur est ensuite divisée par 16 par les circuits internes de l'ACIA. La division de l'horloge par 16 est obtenue en mettant les bits 0 et 1 dans les états suivants :

BIT 0 = 1 **BIT 1 = 0**

La figure binaire 101 pour les bits 4, 3 et 2 impose à l'interface de travailler avec des mots de 8 bits et un seul signal de stop.

L'ACIA, avec cette valeur, n'effectue pas non plus de contrôle de parité.

Les bits 5 et 6 ne servent que lorsque l'ACIA est employé au sein d'un modem.

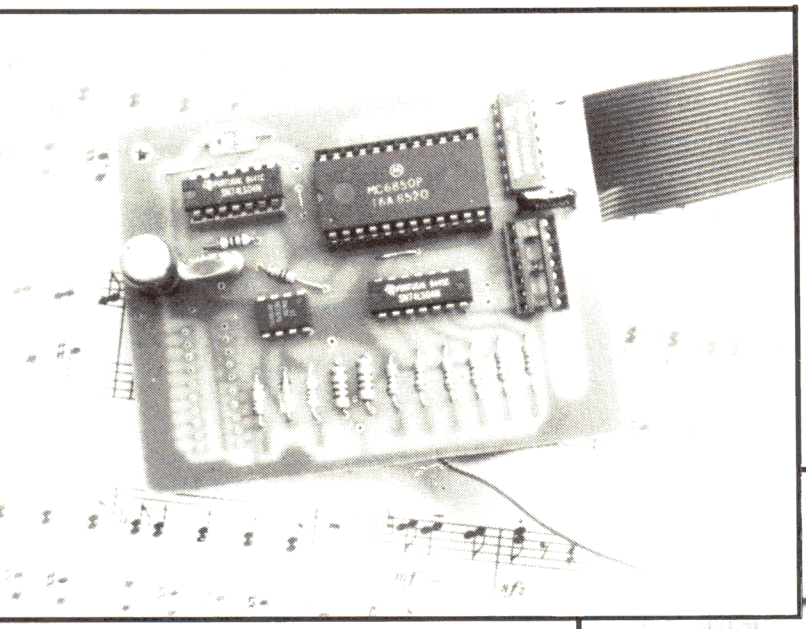
Enfin, le bit 7 de ce registre permet à l'ACIA de demander des interruptions lorsqu'il reçoit des messages MIDI.

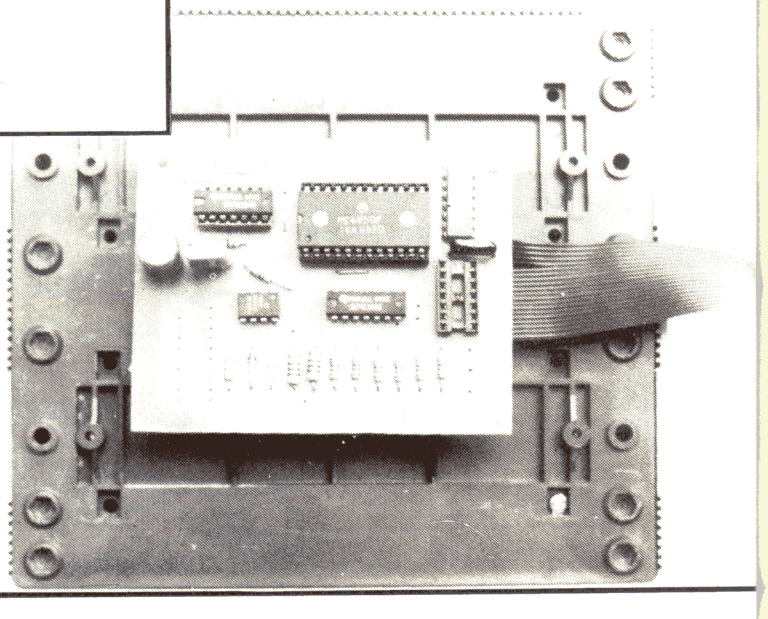
LISTE DES COMPOSANTS

IC 1	SN 74LS04	Sextuples inverseurs TTL LS
IC 2	SN 74LS04	
IC 3	6N 138	Photo-coupleur rapide
IC 4	CD 4040	Compteur-Diviseur C.MOS
IC 5	6850	ACIA
R1 à R9	Résistances 220 Ohms, 1/4 Watt, 10%	
R10	Résistances 270 Ohms, 1/4 Watt, 10%	
R11 à R12	Résistances 1KOhms, 1/4 Watt, 10%	
C1	Condensateur Mylar 0,1 MF, 63 Volts	
C2	Condensateur Tantale 47 MF, 10 Volts	
C3	Condensateur 2,2 NF, 63 Volts.	
Q	Quartz de 2000,00 KHz (soit 2 MHz)	
—	5 prises DIN 5 broches/180°/FEMELLES/Socles	
D1	DIODE au Silicium genre 1N 4148 ou 1N914 (K = Cathode)	
—	Une carte d'étude pour montage électronique, enfichable dans un des slots de l'Apple.	
—	Eventuellement : supports à souder pour les 5 circuits intégrés du montage. ■	

INTERFACE MIDI

Jean-Paul VERPEAUX





Ces photos se rapportent au prototype et ne sont pas rigoureusement conformes au schéma électrique que nous vous présentons page 47.

Marcel COTTINI

LES ROUTINES
d'entrée/sortie d'un caractère

LA ROM MONITOR

La ROM du Monitor comporte un certain nombre de routines et sous-programmes. C'est un langage à part entière qui regroupe des programmes système. Au niveau le plus élémentaire se trouve le microprocesseur qui est associé à un certain nombre de circuits et d'interfaces pour périphériques. Le langage élémentaire du processeur étant le langage machine, ce dernier permet de faire fonctionner les circuits de façon cohérente. Une des routines élémentaires du Monitor nous permet de vérifier, visualiser ou modifier directement le contenu des mémoires ou des registres du processeur. D'autres fonctions du Monitor existent, entre autres les routines d'entrée et de sortie d'un caractère.

L'accès au Monitor à partir de l'AppleSoft se fait en tapant :

CALL - 151 Entrée sans bip dans le Monitor.

ou encore :

CALL - 155 Entrée avec un bip dans le Monitor.

ROUTINES D'ENTRÉE/SORTIE

COUT ($FDED)

Appelle le sous-programme courant de sortie de caractères. Le code ASCII du caractère se trouvera dans l'accumulateur. COUT appelle par la suite le sous-programme dont l'adresse est mémorisée en CSWL-CSWH (adresses $36 et $37). qui est la sortie standard de caractères COUT1.

COUT1 ($FDFØ)

Affiche le caractère présent dans l'accumulateur sur l'écran de l'Apple, à la position courante du curseur qui avance , celui-ci étant lui-même lié à une opération de sortie. Il place le caractère en se servant de l'état de l'adresse du mode normal ou inverse. Il prend en compte les codes des caractères suivants :

- RETURN
- ESPACE
- Saut de ligne
- BELL.

Il laisse tous les registres intacts à son retour.

RDKEY ($FDØC)

Rentrer un seul caractère. C'est le sous-programme d'entrée de caractères. Il place un curseur clignotant à la position courante, et saute à la routine dont l'adresse est dans KSWL-KSWH ($38 et $39). Le plus souvent, c'est la routine standard d'entrée KEYIN qui retourne avec le caractère dans l'accumulateur.

KEYIN ($FD1B)

Entrée de caractères par le clavier, et écho sur l'écran ; le code du caractère est dans l'accumulateur. Il lit le clavier de l'Apple II, attend qu'une touche soit enfoncée, et rend aléatoire le nombre mémorisé aux adresses $4E-$4F. Si une touche est actionnée, KEYIN supprime le curseur clignotant de l'écran et renvoie le code de la touche se trouvant dans l'accumulateur.

GETLN ($FD6A)

C'est le sous-programme standard d'entrée pour

des lignes entières de caractères. Le programme appellera GETLN, après avoir placé le solliciteur (prompt) à l'adresse $33, puis GETLN met la ligne de commande dans la zone tampon d'entrée, et le registre X contient la longueur de la ligne d'entrée au retour du sous-programme.

GETLNZ ($FD67)

Entrer une ligne de commande. C'est un autre point d'entrée pour GETLN, qui enverra un caractère RETURN à la sortie standard, puis continuera dans GETLN.

GETLN1 ($FD6F)

Entrer une ligne de commande sans solliciteur. C'est un autre point d'entrée pour GETLN, qui n'affichera pas de solliciteur avant d'accepter la ligne de commande. Si l'utilisateur annule la ligne de commande à cause d'un trop grand nombre de caractères, ou en tapant CTRL-X, alors GETLN1 utilisera le contenu de l'adresse $33 comme solliciteur quand il entrera une nouvelle ligne.

ROUTINES D'ENTRÉE/SORTIE ET SES PARTICULARITÉS

GETLNZ : entrée ——» $FD67 (64871 ou - 665)

Situation des registres à l'entrée :

Accumulateur	aucune action
Registre X	aucune action
Registre Y	aucune action

Situation des paramètres de la page zéro :

BASL,H	aucune action
CH	aucune action
CV	ligne courante du curseur

Situation au retour de la routine :

Accumulateur	contient le code ASCII
Registre X	contient le nombre de caractères avant RETURN
Registre Y	contient WNDWDTH, qui est le nombre de lignes
CH	contient Ø
CV	contient le numéro de la ligne courante
BASL,H	contient les paramètres CV et WNDWTH

GETLN : entrée ——» $FD6A (64874 ou - 662)

Situation des registres à l'entrée :

Accumulateur	aucune action
Registre X	aucune action
Registre Y	aucune action

Situation des paramètres de la page zéro :

BASL,H	colonne et ligne courante
CV	numéro de ligne qui vient d'être rentrée
CH	numéro de ligne où sera placé le prompt

Situation au retour de la routine :
identique à GETLNZ.

NXTCHAR : entrée ——» $FD75 (64885 ou - 651)

Situation des registres à l'entrée :

Accumulateur	aucune action
Registre X	Ø avant l'entrée de données dans le buffer $ØØ2ØØ (Page $2)
Registre Y	aucune action

Situation des paramètres de la page zéro :

BASL,H	comme CV, dans les limites de la fenêtre
CH	position au moment de l'entrée du caractère au clavier
CV	comme BASL,H dans la limite de la fenêtre

Situation au retour de la routine :
identique à GETLNZ.

CAPTST : entrée ——» $FD7E (64894 ou - 642)

C'est une partie de la routine NXTCHAR. Les registres et paramètres de la page zéro sont identiques à la routine NXTCHAR.

NOTCR : entrée ——» $FD3D (64829 ou - 707)

Situation des registres à l'entrée :

Accumulateur	caractère à afficher, et récupéré par COUT
Registre X	IN,X pointe le dernier caractère
Registre Y	aucune action

Situation des paramètres de la page zéro :

BASL,H	aucune action
CH	aucune action
CV	aucune action

NOTCR1 : entrée —» $FD5F (64863 ou - 673)

N'est pas recommandé pour une utilisation comme routine séparée, à partir du clavier ou d'un programme.

CANCEL : entrée —» $FD62 (64866 ou - 670)

Situation des registres à l'entrée :

Accumulateur	aucune action
Registre X	aucune action
Registre Y	aucune action

Situation des paramètres de la page zéro :

BASL,H	aucune action
CH	aucune action
CV	position de la ligne actuelle

BACKSPC : entrée —» $FD71 (64881 ou - 655)

Situation des registres à l'entrée :
identique à NEXTCHR

Situation des paramètres de la page zéro :
identique à NEXTCHR, une fois que le pointeur du registre X a été mis à jour, suite à la suppression d'un caractère.

RDCHAR : entrée —» $FD35 (64821 ou - 715)

Situation des registres à l'entrée :

Accumulateur	aucune action
Registre X	aucune action
Registre Y	aucune action

Situation des paramètres de la page zéro :

BASL,H	position de la ligne actuelle dans la limite de la fenêtre. Comme CV
CV	ligne actuelle, comme BASL,H
CH	position horizontale du curseur dans les limites de la fenêtre

Situation au retour de la routine :

Accumulateur	contient la valeur de la touche pressée
Registre X	pas de changement
Registre Y	contient la valeur de CH
BASL,H	CV et CH sont altérés si la touche ESC est sollicitée

RDKEY : entrée —» $FD0C (64780 ou - 756)

Situation des registres à l'entrée :

Accumulateur	aucune action
Registre X	aucune action
Registre Y	aucune action

Situation des paramètres de la page zéro :

BASL,H	la position de la ligne traitée actuellement est stockée à cette adresse
CV	identique à BASL,H
CH	position horizontale actuelle du curseur

Situation au retour de la routine :

Accumulateur	contient la valeur du caractère lu au clavier
Registre X	pas de changement
Registre Y	contient la valeur de CH
CV	est utilisé pour calculer la nouvelle ligne
CH	pas de changement
BASL,H	contient maintenant les nouvelles valeurs

KEYIN : entrée —» $FD1B (64795 ou - 741)

Situation des registres à l'entrée :

Accumulateur	valeur du caractère rentré lors du clignotement du curseur
Registre X	aucune action
Registre Y	utilisé pour afficher la valeur de l'accumulateur à la position de BASL,H

Situation des paramètres de la page zéro :

BASL,H	utilisé par les registres X et Y
CH	aucune action
CV	position verticale de la ligne actuelle

Situation au retour de la routine :

Accumulateur	contient la valeur rentrée au clavier (STROBE)

Les autres registres restent inchangés.

ESC : entrée —» $FD2F (64815 ou - 721)

Situation des registres à l'entrée :

Accumulateur	aucune action
Registre X	aucune action
Registre Y	aucune action

Situation des paramètres de la page zéro :

BASL,H	aucune action
CH	aucune action
CV	position verticale de la ligne actuelle

ESCNEW : entrée —» $FBA5 (64421 ou - 1115)

Situation des registres à l'entrée :

Accumulateur	aucune action
Registre X	aucune action
Registre Y	aucune action

Situation des paramètres de la page zéro :

BASL,H	aucune action

(suite page 54)

CH	aucune action
CV	position verticale de la ligne actuelle

ESC1 : entrée —» $FC2C (64556 ou - 980)

Situation des registres à l'entrée :

Accumulateur	aucune action
Registre X	aucune action
Registre Y	aucune action

Situation des paramètres de la page zéro :

BASL,H	aucune action
CH	aucune action
CV	position verticale de la ligne actuelle

Remarque : Le sous-programme CAPTST met automatiquement en affichage majuscule tout caractère rentré à partir du clavier.

Signification des paramètres de la page zéro :

CH	($24)	Position horizontale du curseur. Valeurs entre 0 et 39 incluses.
CV	($25)	Position verticale du curseur. Valeurs entre 0 et 23 incluses.
BASL BASH	($28) ($29)	Adresse de la première colonne de la ligne courante pour la page TEXT (1024 à 2000 ou $400 à $7D0). Octet poids faible (BASL), et octet poids fort (BASH). ■

GRAND CONCOURS VERSION SOFT/APPLE FRANCE

LES FRUITS D'UNE PASSION : 20 PROJETS SÉLECTIONNÉS

Le grand jury du Concours Version Soft/Apple France "Les fruits d'une passion 87" vient de sélectionner les 20 projets qui recevront et ceci jusqu'au 15 septembre le soutien des développeurs Version Soft et Apple.

Dans son intégralité le jury a souligné la diversité, la créativité et surtout la grande qualité des quelques 50 dossiers qui lui ont été proposés.

Pour Neil Minkley Directeur du Développement Stratégique d'Apple France :

- "L'intérêt pour la programmation et notamment pour le développement sur Macintosh et aujourd'hui sur l'Apple *II GS* est toujours plus grand, la diversité et la créativité des projets sélectionnés le montrent largement. Il est intéressant de signaler également le grand nombre de projets n'émanant pas d'informaticiens et ceci avec un très haut niveau technique utilisant des interfaces conviviaux modernes".

Pour Patrick Arnoux, rédacteur en chef de Challenges :

- "Ce concours montre encore une fois la grande créativité française dans le domaine du logiciel et ceci dans tous les secteurs".

Pour Philippe Voisin de Microshop :

- "La passion pour le développement indépendant sur Macintosh continue".

Pour Gérard Verin directeur général de Version Soft :

- "Ce concours démontre la capacité créative des développeurs indépendants, Version Soft est particulièrement fier de leur avoir offert une structure originale pour mener à bien leurs projets".

Rappelons que dans la deuxième phase du grand Concours "Les fruits d'une passion 87" les auteurs des 20 projets sélectionnés recevront jusqu'au 15 septembre un soutien au développement de la part des développeurs Version Soft et Apple.

Ce sont parmi ces 20 projets que seront sélectionnés les 5 gagnants, les prix étant décernés dans le cadre d'Apple Expo fin octobre. ■

François GALLET

LA SOURIS ET L'ASSEMBLEUR

QUATRIÈME PARTIE : Le mouvement du pointeur à l'écran

A ceux qui ont suivi cette série (et que j'entends piaffer d'impatience), je n'ai qu'une chose à dire : cet article sera leur récompense.

En effet, ce dernier module permet enfin de disposer du pointeur sur l'écran. Il suffit de le rajouter au code déjà existant, ainsi que les quelques instructions suivantes :

```
602A :   20 ED 61    JSR  AFFICHE   ; Placement du pointeur
                                      sur l'écran.
60FD :   20 B4 61    JSR  MOUVT     ; Mouvement du pointeur
                                      s'il y a lieu.
6106 :   20 B4 61    JSR  MOUVT     ; ...
610F :   20 B4 61    JSR  MOUVT     ; ...
6118 :   20 B4 61    JSR  MOUVT     ; ...
```

En ce qui concerne les explications du source, il y peu de chose à dire : il vous suffira de vous reporter au listage commenté. Si vous voulez récupérer les coordonnées du pointeur dans l'un de vos programmes en Basic, vous pourrez le faire par :

PEEK (24873) pour l'abscisse
PEEK (24874) pour l'ordonnée.

Pour ceux qui désirent un exemple de programme, voici quelques lignes "parlantes" :

```
10 HOME
20 PRINT CHR$ (4)"BLOAD SOURIS.C"
30 CALL 24576 : REM Activation de la Souris
40 X = PEEK (24873) : Y = PEEK (24874)
50 VTAB 2 : PRINT "Abscisse : ";X; : HTAB 20 :
   PRINT "Ordonnée : ";Y
60 IF X = 40 AND Y = 12 THEN END
70 VTAB 2 : HTAB 11 : PRINT "  ";:HTAB 31 :
   PRINT "  "
80 GOTO 40
```

Une chose importante que j'ai négligé de vous dire : cette série d'articles ne s'adresse bien évidemment qu'aux personnes possédant un APPLE *IIc* ou un APPLE *IIe+*, équipés de la carte Souris. De plus, les routines décrites ne fonctionnent qu'en 80 colonnes. Je pense qu'à l'heure du *IIGS*, beaucoup d'entre vous possèdent déjà cette configuration.

Je vous souhaite une bonne et fructueuse programmation, et je vous dis à bientôt, dans ces colonnes, pour d'autres articles qui vous aideront à mieux maîtriser votre APPLE en Assembleur.

Toujours pour ceux qui ne travaillent que sous Moniteur, commencez par recharger ce qui a été fait jusqu'à présent, écrivez les octets du dernier module, en n'oubliant pas les lignes à insérer décrites plus haut, puis sauvez le tout par :

BSAVE SOURIS.C,A$6000,L545.

Lancez comme auparavant par CALL 24576 ou $6000G. Il est évident que l'adresse de début a été choisie d'une manière empirique, et que vous êtes seuls juges de l'endroit opportun où débutera ce programme de traitement de la Souris, en fonction de vos propres applications (faites tout de même attention aux adresses de branchement des modules entre eux si vous implantez ailleurs qu'en $6000).

François GALLET.

(suite page 56)

```
0  ******************************************************
1  *                                                    *
2  *   Programmation de la Souris : Quatrième partie    *
3  *                                                    *
4  ******************************************************
5  *
6  *
7  BASCALC  EQU  $FBC1       :Calcule l'adresse de base dans l'écran texte
8                            ;d'une ligne dont le numéro est dans l'accu.
9  ETAT     EQU  $77C        :Etat du bouton et des interruptions
10 PAGE1    EQU  $C054       :Commutateur logiciel "Mémoire principale"
11 PAGE1X   EQU  $C055       :Commutateur logiciel "Mémoire auxiliaire"
12 *
13          ORG  $61B4
14 *
34 *
35 *
36 *   Sous programme : MOUVT  (Détecte si la souris à bougée)
37 *   ------------------------------------------------------
38 *
39 MOUVT    LDA  ETAT         :Etat du bouton et des interruptions
40          ASL               ;3 petits décalages à gauche...
41          ASL               ;...pour tester le bit 5...
42          ASL               ;...significatif du déplacement de la souris
43          BCC  BREAK        :C = 0 si pas déplacement...
44          JSR  RESTORE      :C = 1 donc on restaure le caractère occulté
45          JSR  NEWCOORD     :Nouvelles coordonnées...
46          JSR  AFFICHE      ;et affichage de la flèche
47 BREAK    RTS               :Retour d'appel
48 *
49 *
50 *   Sous programme : RESTORE
51 *   ------------------------
52 *
53 RESTORE  LDA  CV           :Sauvegarde de...
54          PHA               ;...la position...
55          LDA  CH           ;...actuelle du...
56          PHA               ;pointeur
57 *
58          LDA  QUOTX        :N° de colonne compris entre 0 et 79
59          JSR  COM.LOG      ;Mémoire principale ou auxiliaire ?
60          LDA  QUOTY        :On charge le numéro de ligne...
61          JSR  BASCALC      ;...et on calcule l'adresse de base écran
62          LDY  COL40        :Numéro de colonne
63          LDA  MEMORY       :On reprend le mème caractère que celui qui a
64                            ;été sauvegardé dans AFFICHE
65          STA  ($28),Y      ;...et on le remet en lieu et place de l'icone
66          BIT  PAGE1        ;On se repositionne en mémoire principale si
67                            ;on n'y était pas déjà
68 *
69 RETOUR   PLA               ;Restauration de...
70          STA  CH           ;...la position...
71          PLA               ;...du pointeur...
```

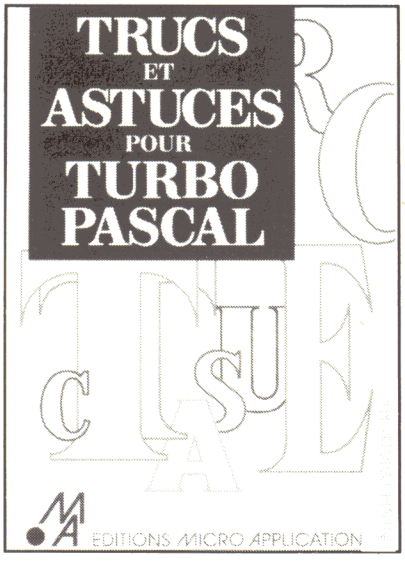
```
 72            STA  CV          ;..et...
 73            JSR  VTABZ       ;placement à l'écran
 74            RTS              ;Retour d'appel
 75  *
 76  *
 77  *    Sous programme : AFFICHE
 78  *    ------------------------
 79  *
 80  AFFICHE   LDA  CV          ;Sauvegarde de...
 81            PHA              ;...la position...
 82            LDA  CH          ;...actuelle du...
 83            PHA              ;pointeur
 84  *
 85            LDA  QUOTX       ;N° de colonne compris entre 0 et 79
 86            JSR  COM.LOG     ;Mémoire principale ou auxiliaire ?
 87            LDA  QUOTY       ;On charge le numéro de ligne...
 88            JSR  BASCALC     ;...et on calcule l'adresse de base écran
 89            LDY  COL40       ;Numéro de colonne
 90            LDA  ($28),Y     ;On prend le caractère présent à cet endroit
 91            STA  MEMORY      ;...et on le sauvegarde provisoirement
 92            LDA  £$42        ;Code de l'icone "flèche"
 93            STA  ($28),Y     ;...que l'on affiche à la place du caractère
 94                             ;sauvegardé
 95            BIT  PAGE1       ;On se repositionne en mémoire principale si
 96                             ;on n'y était pas déjà
 97  *
 98            BRA  RETOUR      ;Terminé...
 99  *
100  *
101  *    Sous programme : COM.LOG (commute en PAGE 1 si la colonne est impaire
102  *    ------------------------
103  *    ou en PAGE 1X si elle est paire et transforme le numéro de colonne en
104  *    40 colonnes .
105  *
106  COM.LOG   ROR              ;Une rotation à droite pour tester le bit de
107                             ;poids le plus faible (bit de droite)
108            BCS  MEM         ;S'il vaut 1 , le nombre est impair --> Page 1...
109            BIT  PAGE1X      ;S'il vaut 0 , le nombre est pair --> Page 1X...
110            BRA  CONT4       ;...et on saute la ligne qui suit
111  MEM       BIT  PAGE1       ;Commutation en Page 1 si nombre impair
112  CONT4     ROL              ;On restitue le bit de droite...
113            LSR              ;...on décale à droite pour diviser par 2
114            STA  COL40       ;et on stocke le résultat (en 40 colonnes)
115            RTS              ;Retour d'appel
```

```
BSAVE S4,
A$61B4,L109
```

```
61B4 : AD 7C 07 0A 0A 0A 90 09 20 C6 61 20 63 61 20 ED   DE1F
61C4 : 61 60 A5 25 48 A5 24 48 AD 29 61 20 10 62 AD 2A   7D84
61D4 : 61 20 C1 FB AC 23 61 AD 26 61 91 28 2C 54 C0 68   3B02
61E4 : 85 24 68 85 25 20 22 FC 60 A5 25 48 A5 24 48 AD   9F29
61F4 : 29 61 20 10 62 AD 2A 61 20 C1 FB AC 23 61 B1 28   9539
6204 : 8D 26 61 A9 42 91 28 2C 54 C0 80 D3 6A B0 05 2C   E796
6214 : 55 C0 80 03 2C 54 C0 2A 4A 8D 23 61 60            D5BD
```

Votre bibliothèque INFORMATIQUE

par NESTOR



• TRUCS ET ASTUCES POUR TURBO PASCAL (SGONINA WARNER)

Bien des possesseurs de PC et compatibles connaissent le langage TURBO PASCAL et l'apprécient. Gageons qu'ils puiseront volontiers dans cet excellent bouquin de programmation dont ils expérimenteront les nombreux conseils, méthodes, programmes et utilitaires.

Les principaux domaines de la programmation y sont traités avec clarté : tri, affichage, saisie, accès disque, date et heure sous MS/PC-DOS.

A noter que les lecteurs peuvent obtenir une disquette contenant tous les programmes et utilitaires du livre (120 F).

258 pages — 149 F TTC

***Micro Application** 13, rue Sainte Cécile — 75009 PARIS*

• LE LIVRE DU G.W BASIC ET P.C BASIC (Bomanns)

Si vous avez envie de programmer en GW-BASIC, laissez-vous guider pas à pas par l'auteur de ce manuel DATA BECKER.

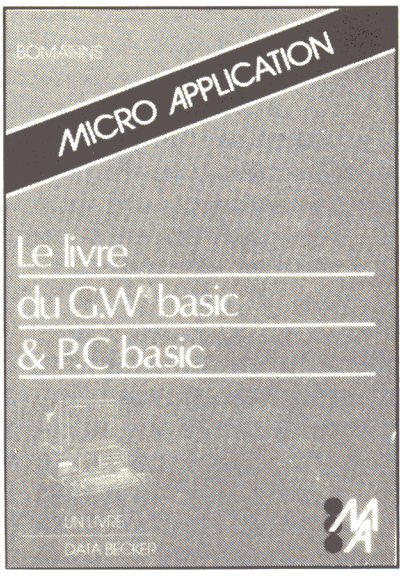
Comme le souligne son texte de présentation, c'est une introduction pour le débutant et un aide-mémoire pour l'utilisateur. On pourrait se contenter d'y survoler les nombreuses et intéressantes possibilités du langage, mais on va plus loin. En effet, après avoir expliqué les instructions et les fonctions, les variables, la gestion des fichiers et les commandes d'imprimante, on aborde des sujets au moins aussi intéressants :

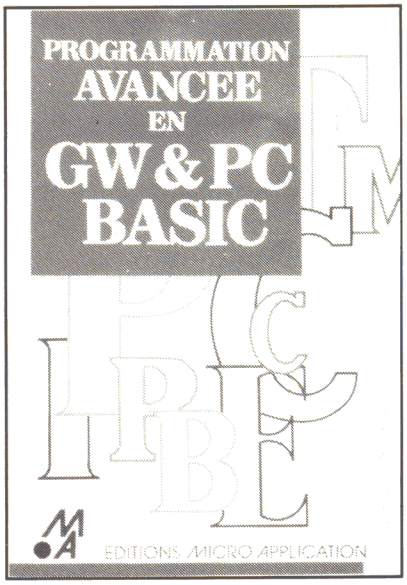
— La programmation sonore.
— Le graphisme et la technique des fenêtres.
— Les interfaces.
— La programmation par interruptions.

La compilation de programmes en GW-BASIC n'est pas négligée et pas davantage les liens entre le langage et MS-DOS. Plusieurs petits utilitaires sont fournis en prime... avant un index agréable à consulter.

328 pages — 149 F TTC

***Micro Application** 13, rue Sainte Cécile — 75009 PARIS*





• PROGRAMMATION AVANCÉE EN G.W & PC BASIC (Bomanns)

Suite logique du précédent, ce manuel s'adresse à des programmeurs plus avertis, désirant concevoir des logiciels de qualité professionnelle.

On retrouve la même clarté que dans le premier, mais les exemples y sont beaucoup plus nombreux.

Le lecteur saura comment redéfinir des touches d'édition et de fonction, bénéficiera d'une routine d'entrée universelle ainsi que d'une adaptation universelle de l'imprimante.

Les vecteurs d'interruptions de MS/DOS-BIOS n'auront plus de secrets pour lui et il saura (presque) tout des instructions et fonctions du compilateur QUICK BASIC.

450 pages — 199 F TTC.

***Micro Application** 13, rue Sainte Cécile — 75009 PARIS*

DIDACTIQUE

Yvan KOENIG

Rendre Lam au GS !

Sans doute avez-vous déjà constaté que la bonne vieille routine de S.H. LAM (qui permet de mettre en place des routines HEX depuis le Basic) ne fonctionne pas toujours sur le GS. Sur un court programme, il se peut que l'on ne remarque rien, mais en général, on provoque un message d'erreur montrant à l'habitué qu'il existe un problème du côté de la pile. Le fouineur que je suis ne pouvait se contenter de cette amère constatation.

J'ai donc décidé de partir aux nouvelles et, dans ce cas, rien ne remplace une routine qui sauve la pile AVANT et la pile APRÈS. Avec ça, j'ai déjà dépanné des collaborateurs américains de NIBBLE et Call A.P.P.L.E. Il était temps que je me rende le même service !

Sauvez la petite routine GSLAM et la DÉMO BLAM, puis faites RUN BLAM. Si tout se passait correctement, la routine de LAM s'exécuterait 2 fois mais, on obtient un magnifique *NEXT WITHOUT FOR IN 100*. Pourtant, tout est bien à sa place. Faites un nouveau RUN.

Vous pouvez retrouver :

page zéro AVANT en 4000 . . . , APRÈS en 4300 . . .
pile AVANT en 4100 . . . , APRÈS en 4400 . . .
inbuf AVANT en 4200 . . . , APRÈS en 4500 . . .

Vous allez pouvoir vous rendre compte que le retour se fait avec un pointeur de pile erroné. En $41FE vous avez le pointeur de pile AVANT (E4) et, en $44FE, le même pointeur après, ce devrait être le même mais, bien entendu, ça ne l'est pas (CC).

Regardons de près :

en 41E5/6 on trouve l'adresse D823-1, c'est l'adresse utile de la routine de LAM (fin de la chaîne).

en 41E7 on a $81 c'est le token FOR.
41E8/9 0956 c'est l'adresse de la variable I indice de boucle.
41EA/E 81 80 00 00 00 c'est le pas 1 (step) en format flottant.
41EF 01 c'est le signe de step, ici +.
41F0/4 82 00 00 00 00 c'est la limite (2) de la boucle.
41F5/6 0014 c'est le num de la ligne FOR (ici 20).
41F7/8 0850 c'est l'adresse de fin de la ligne 20.

Le pointeur de pile étant incorrect en retour du CALL-144, APPLESOFT ne retrouve pas ces informations et nous en informe brutalement avec son *NEXT WITHOUT FOR*.

Maintenant que nous savons ce qui se produit, le remède est simple (j'élimine bien entendu la formule fer à souder/programmeur d'Eproms).

Effacez les lignes 45 et 75 puis RUN. Cette fois tout se passera bien. CALL 768 aura sauvé le pointeur de pile, CALL 773 l'aura rétabli.

Essayez, dans le programme de patch du DOS publié dans *Tremplin Micro n°14*, de remplacer tous les CALL-144 par CALL768:CALL-144: CALL773 et ajoutez la ligne 74 qui met en place les 10 octets qui sauvent et récupèrent S.

```
74 FOR I = 768 TO 77: READJ:POKEI,J:
NEXT:DATA186,142,251,1,96,174,251,1,154,96
```

Si vous avez la disquette *Tremplin Micro n°16*, transférez le fichier DOSYK sur votre disquette DOS 3.3 favorite, c'est nettement moins fatigant.

Le programme fonctionnera parfaitement sur votre GS. Bien entendu, le programme ainsi modifié fonctionnera aussi sur un *IIe* ou un *IIc*.

SHLAM

```
10  REM ********
15  REM * BLAM *
20  REM ********
25  PRINT  CHR$ (4)"BLOAD GSLAM"
30  FOR PASS = 1 TO 2
35 HEX$ = "1000:4C 58 FF"
40 HEX$ = HEX$ + " N D823G": FOR
     I = 1 TO  LEN (HEX$): POKE 5
     11 + I, ASC ( MID$ (HEX$,I,1
     )) + 128: NEXT : POKE 72,0
45  GOTO 60: REM "provisoire, sau
     te le remède
50  CALL 768
60  CALL 778: REM "copie 3 pages
70  CALL  - 144
75  GOTO 90: REM "provisoire, sau
     te le remède
80  CALL 773
90  CALL 782: REM "copie 3 pages
95  NEXT
```

```
                        1
                        2    ********************************
                        3    *       rendre LAM au GS       *
                        4    *                              *
                        5    * Yvan KOENIG         18/06/87 *
                        6    ********************************
                        7
                        8    UNUSED   =     $1FB        ;Adresse inemployée en pile
                        9                               ; (sous APPLESOFT)
                       10
                       11            ORG    $300
                       12
0300: BA                13    CALL768  TSX
0301: 8E FB 01          14             STX    UNUSED     ;Sauve pointeur de pile
0304: 60                15             RTS
                       16
0305: AE FB 01          17    CALL773  LDX    UNUSED
0308: 9A                18             TXS               ;Restaure pointeur de pile
0309: 60                19             RTS
                       20
030A: A2 40             21    CALL778  LDX    £)$4000
030C: D0 02             22             BNE    DOSAVE
030E: A2 43             23    CALL782  LDX    £)$4300
                       24
0310: 8E 25 03          25    DOSAVE   STX    :2+2
0313: E8                26             INX
0314: 8E 2B 03          27             STX    :3+2
0317: E8                28             INX
0318: 8E 31 03          29             STX    :4+2
031B: BA                30             TSX
031C: 8E FB 01          31             STX    UNUSED     ;On pourra le retrouver
                       32                               ;en 41FB (avant) ou 44FB
031F: A2 00             33             LDX    £0          (après)
0321: B5 00             34    :1       LDA    $00,X
0323: 9D 00 40          35    :2       STA    $4000,X    ;Sauve Page zéro
0326: BD 00 01          36             LDA    $100,X
0329: 9D 00 41          37    :3       STA    $4100,X    ;Sauve la pile
032C: BD 00 02          38             LDA    $200,X
032F: 9D 00 43          39    :4       STA    $4300,X
0332: E8                40             INX
0333: D0 EC             41             BNE    :1
0335: 60                42             RTS
```

Yvan KŒNIG

Copie de disque 3"5 dans /RAM5

Guy-Hachette m'a fait part de demandes de lecteurs souhaitant copier aisément un disque 3"5 entier sur le disque virtuel /RAM5 de leur GS (bien entendu, cela suppose que la carte d'extension soit équipée de 1M0).

Je suppose que ces demandes émanaient de lecteurs ne souhaitant pas utiliser PRODOS16 et MOUSEDESK dont les temps de chargement sont décourageants. Je rappelle que si PRODOS16 est indispensable pour accéder à la totalité des outils du GS, il est parfaitement possible de travailler sans lui. Dans ce cas, le nombre d'outils réellement exploitable est faible, mais on dispose d'un équipement "rapide" et de la quasi totalité des "anciens" logiciels.

Possédant une certaine expérience dans le domaine des commandes externes ProDOS, j'ai essayé de résoudre le problème posé en créant une nouvelle commande que j'ai baptisée DCOPY. J'ai bien entendu utilisé MERLIN.PRO, mon assembleur favori (en 8 bits, puisqu'à ce jour, en 16 bits, nous n'avons le choix qu'entre ORCA.M et son grand frère APW assembler).

Précisons d'entrée qu'il n'est pas question ici de vous donner un outil de *piratage :* la commande DCOPY ne sait copier que des disquettes normales, dont tous les fichiers seraient accessibles par les commandes classiques de ProDOS.

Le disque virtuel sera supposé dimensionné à 800K lors de la mise en route de la machine. Si tel n'était pas le cas, il faudrait activer le tableau de bord par CTRL pomme ESC et spécifier *taille mini = taille maxi = 800K.* Ensuite, retour au BASIC, extinction de la machine, attente de quelques secondes, puis remise sous tension.

Le programme comporte deux modules. Le premier vous est déjà connu, c'est RELOPRO dont la dernière mouture vous a été présentée dans *Tremplin Micro n°14* (SONSIR). Le fichier source ne sera pas reproduit ici.

Le second est le module actif. Il commence par une vingtaine d'instructions qui assurent la compatibilité avec les commandes ProCMD de Glen Bredon. Ensuite, la commande envoyée est analysée. Si ce n'est pas DCOPY, on renvoie le bébé à BASIC.SYSTEM. Si nous sommes bien en présence de DCOPY, on laisse BASIC.SYSTEM analyser la suite qui peut être :

1. *(,Ssource ,Dsource)* pour transmettre la localisation du disque source, dans le cas du transfert en direction du disque virtuel ;
2. *(R ,Sdest Ddest)* pour indiquer la localisation du disque récepteur, dans le cas où l'on souhaite transférer le contenu du disque virtuel sur un disque magnétique. Il est alors *INDISPENSABLE que le disque ait été préalablement formaté.* Le *(R)* qui suit immédiatement *DCOPY* dans ce cas peut être remplacé par n'importe quelle chaîne de caractères assimilable à un titre de fichier puisque seule la présence d'un *titre* est significative pour DCOPY. Pas de *titre* et l'on copie dans /RAM5 ; un *titre* et l'on récupère /RAM5.

Lorsque l'analyse est terminée, BASIC.SYSTEM rend le contrôle à notre routine en TRUECMD. Si l'option S/D n'a pas été employée, on s'arrête sur un "SYNTAX ERROR". Attention ! l'analyse effectuée par BASIC.SYSTEM ne permet pas de vérifier que l'on a communiqué les deux paramètres. *(Suite page 62)*

Il serait possible de reprogrammer l'analyse de la commande, mais il me semble que ça n'apporterait aucune garantie supplémentaire. Si l'on est capable d'oublier l'un des paramètres, on est capable de se tromper en définissant la valeur de ces paramètres. Sachez d'ailleurs que par défaut, ce sont les valeurs correspondant au disque actif qui seront employées. Donc, si vous travaillez depuis S5D1, c'est ce disque qui sera utilisé. Ceux que cela chagrine pourront se reporter à POM'S 29 où Patrice NEVEU traitait le problème de l'analyse complémentaire.

Quelques pointeurs sont initialisés et l'on copie tranquillement les 1600 blocs un par un. C'est ringard ! vont clamer les esprits forts. Ils auront tort, et j'ai eu tort avant eux. J'ai cru qu'il serait intéressant de travailler sur des groupes de blocs et non sur des blocs isolés. Hélas ! un par un, seize par seize ou même trente-deux par trente-deux, le temps de copie reste le même, environ 1mn40s. J'ai également comparé avec le temps nécessaire pour copier avec COPYII PLUS : ce dernier est un peu plus long, mais c'est dû au fait que ce logiciel affiche les numéros des pistes qu'il manipule. Ceux qui ne me croient pas peuvent m'envoyer 30 francs. Je leur adresserai la version améliorée qui a l'insigne mérite d'utiliser deux pages mémoire au lieu d'une pour un résultat identique.

Lorsque la copie est terminée, il est possible de configurer le tableau de bord en boot sur RAM disk ce qui permet, en cas de besoin, de redémarrer très rapidement.

Deux remarques au sujet des commandes externes.

1° Worth et Lechner ont présenté dans Beneath Apple ProDOS une commande externe TYPE dont le code utile commence par un CLD accompagné du commentaire *identificateur pour BI* (pour les néophytes, BI c'est BASIC.SYSTEM qui est en fait un COMMANDS.INTERPRETER, CI pour APPLE. Ouf !) et depuis, on voit publier des commandes externes de deux types. Celles dont l'auteur a pris la commande TYPE comme exemple, aggravant parfois la situation avec le commentaire *Obligatoire pour ProDOS*, et celles écrites par des émules de St-Thomas qui ont tenu à se reporter aux sources. Ceux-là savent que ce n'est pas pour les commandes externes qu'un CLD est nécessaire (cf. note technique ProDOS £8) mais pour les routines QUIT qui doivent contenir cette instruction en $D100 (cf. note technique ProDOS £14). Ce n'est pas bien grave, mais il est parfois bon d'essayer de stopper la propagation des rumeurs.

2° Dans la note technique ProDOS £8, il y a un bug sur lequel je crois utile de m'arrêter. Afin de demander au BI d'analyser les paramètres Slot et Drive, ils codent : LDA £0, STA PBITS, LDA £4, STA PBITS. Admettons qu'il s'agisse d'une coquille (il y en a d'autres dans la même page) et lisons STA PBITS+1 pour la dernière instruction.
BI, bien brave, va lire la commande mais, puisque PBITS est nul, il ne décodera AUCUN paramètre (faites $A6CB L si vous ne me croyez pas). Pour que BI fasse l'analyse des paramètres, il est INDISPENSABLE que PBITS soit différent de zéro. Personnellement j'ai choisi, dans tous les cas où je n'ai pas d'action particulière à coder dans PBITS, de faire PBITS=$10 (nom de fichier optionnel) ce qui force l'analyse sans effet parasite.

Je me propose de vous offrir, dans un prochain numéro, un petit programme MENU.SYSTEM qui vous permettra de lancer les différentes applications que vous pouvez avoir placées sur votre disquette. Pour ma part, j'y ai mis :

- APPLEWRITER (version patchée),
- MERLIN, PROCMD, COPYII PLUS, SYSUTIL, PROZAP...
- et quelques autres gâteries entre lesquelles je me promène aisément.

Il me faut préciser que SYSUTIL.SYSTEM et les 3 fichiers associés constituent la version GS des bons vieux Utilitaires Systèmes du *IIc* et que c'est bien agréable lorsque l'on ne souhaite pas mettre PRODOS16 sur ses disques (ce qui interdit l'emploi de MouseDesk). Les lecteurs intéressés peuvent en demander (poliment S.V.P.) copie à leur distributeur officiel.
En cas de refus, contactez le support technique ou le Club Apple. ■

```
                        2   ************************************************************
                        3   *                                                          *
                        4   *        K.DCOPY.S  utilise  RELOPRO.S et DCOPY.S          *
                        5   *        ----------------------------------------          *
                        6   *        PRINT CHR$(4)"DCOPY ,S source, D source"          *
                        7   *          copie un disque 3"5 complet en /RAM5            *
                        8   *        ----------------------------------------          *
                        9   *         PRINT CHR$(4) "DCOPYR ,S dest, D dest"           *
                        10  *        copie /RAM5 sur la disquette Sdest,Ddest          *
                        11  *                          préalablement formatée          *
                        12  *                                                          *
                        13  ************************************************************
                        14  * Yvan KOENIG                                   le 12/06/87  *
                        15  ************************************************************
                        16  HASADRS  =     0          ;1 = table adresses,   0 sinon *
                        17  HASDATAS =     1          ;1 = table DATAs,      0 sinon *
                        18  EXTERNE  =     1          ;1 = commande externe, 0 sinon *
                        19  PROTEGE  =     0          ;1 = on protège la routine     *
                        20  INIROUT  =     0          ;1 = on saute dans la routine  *
                        21  ************************************************************
                        22  DISK     KBD   "DISK  (0=non    1=obj/disque) "
                        23  ************************************************************
                        27           PUT   DCOPY
                        1
                        2   *****************************************
                        3   *                                       *
                        4   * DCOPY  ,Ss,Dd copie 3"5 SsDd dans /RAM5 *
                        5   *                                       *
                        6   * DCOPYR ,Ss,Dd copie /RAM5 dans 3"5 SsDd *
                        7   *                              déjà formaté *
                        8   *                                       *
                        9   * Yvan KOENIG                   le 12/06/87 *
                        10  *****************************************
                        11
                        12  CMD_ID   =     $EA
                        13  MEMSIZ   =     $73        ;&74
                        14  CHRGOT   =     $B7
                        15
                        16  ******************************************************
                        17
6200: 4C 1E 62          18  DOSENTRY JMP   PARSE
                        19  NEXTCMD
6203: 4C 9E BE          20  DOSEXIT  JMP   XRETURN
6206: 4C 58 FF          21  AMPEXIT  JMP   IORTS
                        22
6209: D0 FB             23  AMPENTRY BNE   AMPEXIT
620B: A0 00             24           LDY   £0
620D: B9 C9 62          25  :1       LDA   CMDNAME,Y
6210: 20 ED FD          26           JSR   COUT
6213: C8                27           INY
6214: C0 23             28           CPY   £AMPEND-CMDNAME+1
6216: 90 F5             29           BCC   :1
6218: 20 B7 00          30           JSR   CHRGOT
621B: F0 E9             31           BEQ   AMPEXIT    ;Toujours
                        32
```

```
621D: EA          33  IDBYTE   DFB   CMD_ID      ;Provisoire
                  34
621E: A2 04       35  PARSE    LDX   £CMDEND-CMDNAME-1
6220: 8E 52 BE    36           STX   XLEN
6223: E8          37           INX
6224: BD FF 01    38  §L       LDA   IN-1,X
6227: 29 DF       39           AND   £$DF        ;min->MAJ
6229: 5D C8 62    40           EOR   CMDNAME-1,X
622C: 38          41           SEC
622D: D0 D4       42           BNE   DOSEXIT     ;pas MSLOT
622F: CA          43           DEX
6230: D0 F2       44           BNE   §L
6232: 8E 53 BE    45           STX   XCNUM       ;ici X=0
6235: A9 11       46           LDA   £%00010001  ;Nom de Fichier optionnel
6237: 8D 54 BE    47           STA   PBITS
623A: A9 04       48           LDA   £%00000100  ;Autorise Slot Drive
623C: 8D 55 BE    49           STA   PBITS+1
623F: A9 4C       50           LDA   £<TRUECMD
6241: 8D 50 BE    51           STA   XTRNADR
6244: AD 46 62    52           LDA   HIBIT
                  53  HIBIT    =     *-1
6247: 8D 51 BE    54           STA   XTRNADR+1
624A: 18          55           CLC
624B: 60          56           RTS
                  57
624C: AD 57 BE    58  TRUECMD  LDA   FBITS+1
624F: 29 04       59           AND   £%00000100
6251: D0 04       60           BNE   OK
                  61
6253: A9 10       62  ERRSYNT  LDA   £$10        ;SYNTAX ERROR
6255: 38          63           SEC
6256: 60          64           RTS
                  65
6257: AD 62 BE    66  OK       LDA   VDRIV       ;000000Dd (00000001 ou 00000010)
625A: 29 02       67           AND   £%00000010  ;000000D0
625C: 0A          68           ASL               ;00000D00
625D: 0A          69           ASL               ;0000D000
625E: 0D 61 BE    70           ORA   VSLOT       ;0000Dslo
6261: 0A          71           ASL               ;000Dslo0
6262: 0A          72           ASL               ;00Dslo00
6263: 0A          73           ASL               ;0Dslo000
6264: 0A          74           ASL               ;Dslo0000
6265: 8D C7 62    75           STA   UNITSRC
6268: A9 D0       76           LDA   £%11010000  ;Slot5, Drive2
626A: 8D C8 62    77           STA   DESTUNIT
626D: AD 56 BE    78           LDA   FBITS
6270: 4A          79           LSR               ;Y a-t-il un nom de fichier ?
6271: 90 0C       80           BCC   :4          ;=> NON
6273: AD C7 62    81           LDA   UNITSRC     ;On échange pour transfert
6276: AE C8 62    82           LDX   DESTUNIT    ; /RAM5 -> disquette
6279: 8E C7 62    83           STX   UNITSRC
627C: 8D C8 62    84           STA   DESTUNIT
                  85  :4
                  86
```

```
627F: A5 74      87   COPYONE  LDA   MEMSIZ+1
6281: 8D C4 62   88            STA   DATABUF+1  ;Utilise buffer à tout faire
6284: A9 00      89            LDA   £0
6286: 8D C5 62   90            STA   NUMBLOC
6289: 8D C6 62   91            STA   NUMBLOC+1
                 92
628C: AD C7 62   93   :1       LDA   UNITSRC
628F: 8D C2 62   94            STA   UNIT        ;Numéro de l'unité ORIGINE
6292: 20 00 BF   95            JSR   MLI
6295: 80         96            HEX   80          ; Read_Bloc
6296: C1 62      97            DA    TABLE
6298: B0 24      98            BCS   ERREXIT     ;Oops, y-a un os
629A: AD C8 62   99            LDA   DESTUNIT
629D: 8D C2 62   100           STA   UNIT        ;Numéro de l'unité DESTINATION
62A0: 20 00 BF   101           JSR   MLI
62A3: 81         102           HEX   81          ; Write_Bloc
62A4: C1 62      103           DA    TABLE
62A6: B0 16      104           BCS   ERREXIT     ;Oops....
62A8: EE C5 62   105           INC   NUMBLOC     ;Prépare pour bloc suivant
62AB: D0 03      106           BNE   :2
62AD: EE C6 62   107           INC   NUMBLOC+1
62B0: AD C5 62   108  :2       LDA   NUMBLOC
62B3: C9 40      109           CMP   £<1600      ;A-t-on copié tous les blocs
62B5: AD C6 62   110           LDA   NUMBLOC+1
62B8: E9 06      111           SBC   £>1600
62BA: 90 D0      112           BCC   :1          ;Bloc suivant
62BC: 18         113           CLC
62BD: 60         114           RTS
                 115
62BE: 4C 8B BE   116  ERREXIT  JMP   BADCALL
                 117
                 118  ENDPROG  =     *
                 119  ********************************
                 120  ENDADRS  =     *
                 121  ********************************
                 122
62C1: 03         123  TABLE    DFB   3
62C2: 60         124  UNIT     HEX   60
62C3: 00 00      125  DATABUF  HEX   00,00
62C5: 00 00      126  NUMBLOC  HEX   00,00
                 127
62C7: 00         128  UNITSRC  HEX   00
62C8: 00         129  DESTUNIT HEX   00
                 130
62C9: C4 C3 CF   131  CMDNAME  ASC   "DCOPY"
62CC: D0 D9
62CE: A0 BC D2   132  CMDEND   ASC   " <Reverse>,S source, D source"
62D1: E5 F6 E5 F2 F3 E5 BE AC
62D9: D3 A0 F3 EF F5 F2 E3 E5
62E1: AC A0 C4 A0 F3 EF F5 F2
62E9: E3 E5
62EB: 8D         133  AMPEND   HEX   8D
                 134  ********************************
                 135  LAST     =     *-1
```

```
136  ENDALL   =     *                    148  BADCALL  =     $BE8B
137  ********************************    149  XRETURN  =     $BE9E
138                                      150
139  IN       =     $200                 151  COUT     =     $FDED
140                                      152  IORTS    =     $FF58
141  XTRNADR  =     $BE50                153
142  XLEN     =     $BE52                154  MLI      =     $BF00
143  XCNUM    =     $BE53                29   ********************************
144  PBITS    =     $BE54                30            DO    DISK
145  FBITS    =     $BE56                31            SAV   DCOPY
146  VSLOT    =     $BE61                32            FIN
147  VDRIV    =     $BE62                33   ********************************

                --End assembly, 620 bytes, Errors: 0
```

• TECHNIQUES DE PROGRAMMATION EN C

Graphisme et gestion d'écran (par Michel Laurent)

Cet ouvrage présente les sources des fonctions graphiques fondamentales et des fonctions de gestion d'écran en mode alphanumérique, fonctions qui viendront enrichir la bibliothèque standard du compilateur.

Deux chapitres traitent des aspects plus complexes de la programmation. L'un porte sur les notions de multifenêtrage et de menus déroulants avec plusieurs programmes pouvant être utilisés sous la forme proposée ou être intégrés à d'autres programmes d'application. L'autre explique comment réaliser, à partir de la bibliothèque graphique, des programmes d'animation et de composition d'images.

Chaque programme ou liste de fonctions sources est abondamment commenté et illustré. Les exemples élaborés sous MS-DOS sont ainsi facilement transposables à d'autres environnements.

Livre broché de 306 pages — 248 F.

L'auteur : Michel Laurent est titulaire d'un doctorat es-sciences et chargé de recherche au C.N.R.S. Il a développé de nombreux programmes d'applications scientifiques et est coauteur de plusieurs ouvrages parus aux éditions SYBEX.

COMMUNIQUE DE PRESSE

JOURNÉE NATIONALE PORTE OUVERTE MICROTEL — Le 26 septembre 1987

Les 250 clubs MICROTEL ouvrent leurs portes au public le 26 septembre de 10h à 18h ; ce sera l'occasion de découvrir leurs dernières réalisations en informatique, en télématique (création d'un serveur par exemple), en robotique, de connaître les cours qu'il est possible de suivre et de rencontrer les responsables de l'association.

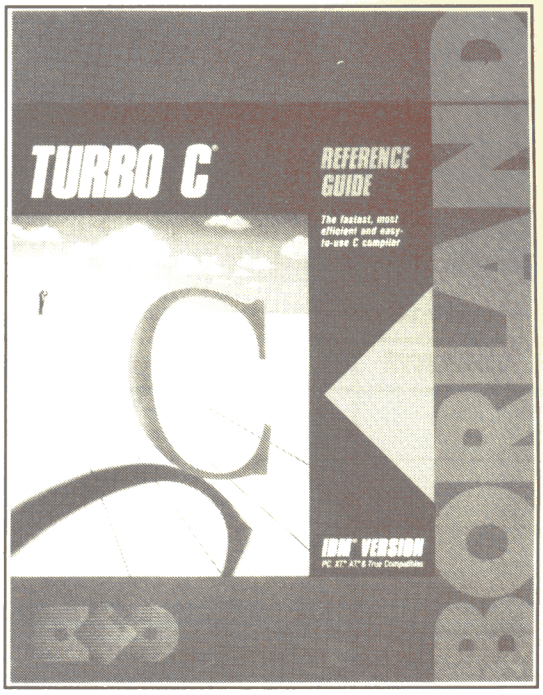
MICROTEL c'est avant tout une structure existant depuis maintenant près de 10 ans et qui reste la seule organisation nationale de clubs d'informatique en France, à laquelle plus de 50 000 personnes ont adhéré.

Pour connaître la liste des clubs MICROTEL : Fédération Nationale MICROTEL — 9, rue Huysmans — 75006 PARIS — Tél. : (16-1) 45-44-70-23/24 ou directement sur minitel 36-14 code ADEMIR.

UN PRODUIT POUR PC ET COMPATIBLES

C... TRÈS FORT !

avec BORLAND



Si vous avez choisi de vous initier au langage **C** en lisant les articles de Claude AUBRY, vous avez sans doute été amené à vous offrir le **C** disponible sur le GS, celui que les développeurs obtiennent en même temps que l'énorme documentation distribuée par VIF.

Ce **C**-là est certes un bon produit, mais on ne peut pas dire qu'il brille par sa rapidité. Ah ! si seulement Borland pouvait s'intéresser au GS et nous fournir un logiciel similaire au Turbo **C** ... disponible sur PC, notre vie, assurément changerait. TURBO **C** est un compilateur de programmes en langage **C** et procure au programmeur toutes les facilités que celui-ci est actuellement en droit d'attendre d'un tel produit : convivialité avec menus déroulants et rapidité grâce à sa technique monopasse.

Mais oui, Turbo **C** ne s'amuse pas — et ne nous amuse pas davantage ! — à créer des fichiers intermédiaires temporaires sur disque, d'où un gain de temps considérable : environ 7000 lignes à la minute sur PC tournant à 6 MHz. Pas mal, non ?

Avec ce nouveau produit de Borland, on dispose d'abord d'un éditeur pleine page comprenant la plupart des fonctions d'un bon éditeur : lecture, écriture, copie, effacement, déplacement de n lignes. Cela se passe dans une fenêtre tandis qu'une autre affiche les éventuels messages du logiciel (erreurs et remarques diverses). C'est à partir de cette seconde fenêtre qu'il est possible d'activer ou non l'affichage des messages et l'indentation automatique. Le passage d'une fenêtre à l'autre est on ne peut plus simple. On notera avec intérêt que la mémoire est modulable, et cela en standard. Il en va de même pour diverses options, mais nous n'avons testé le produit que sur un compatible AT. La documentation, très complète, pourrait dispenser de l'achat d'autres ouvrages si elle était en français... ce qui n'est pas le cas. Reste qu'elle se révèle réellement TRÈS COMPLÈTE. Gageons que, le succès aidant, elle ne tardera pas à être proposée en français. TURBO **C** utilise le standard ANSI **C** et se conforme à la norme IEEE pour les opérations en virgule flottante. Avec ce produit performant, vendu à un prix abordable (moins de 1300 F) et accessible aux non-initiés, Borland met un atout supplémentaire dans son jeu et donne un bon coup de pouce au langage **C** ... sur PC.

BORLAND ET LA FORMATION

Turbo Training est un centre de formation et de distribution exclusivement consacré aux produits *Borland*. On peut y apprendre à programmer en *Turbo Pascal, Turbo Basic,* développer ses propres applications sous *Reflex l'Analyste* ou s'initier à l'intelligence artificielle avec *Turbo Prolog :* la vocation de *Turbo Training* est de couvrir toute la gamme des produits *Borland*.

AU MENU : L'utilisateur suit une formation classique d'une à deux journées lui permettant, soit de s'initier à un produit *Borland* jusque là inconnu, soit d'approfondir ses connaissances d'un logiciel déjà utilisé. La durée et teneur du cours dépendent alors du niveau de l'utilisateur. La formation repose sur des exemples pratiques ; elle est assurée par des enseignants de haut niveau et s'effectue en petits groupes (huit personnes, au maximum).

À LA CARTE : C'est un concept tout nouveau qui offre à l'utilisateur de programmes *Borland,* la possibilité de se rendre chez *Turbo Training* avec ses fichiers, d'exposer un problème particulier et de le résoudre à l'aide d'experts présents sur place. L'utilisateur sait enfin vers qui se tourner lorsqu'il est confronté à un problème qui lui semble insoluble.

Cette formation est assurée de manière individuelle, sous forme de travaux dirigés, tous les jours de 17h à 20 h (du lundi au vendredi). Ce créneau horaire a l'avantage d'être accessible à tout le monde (étudiants, particuliers, personnes travaillant en entreprises). Enfin, *Turbo Training* a décidé d'offrir aux personnes intéressées, une démonstration gratuite sur le produit *Borland* de leur choix ; celle-ci aura lieu tous les jours à partir de 17h, uniquement sur rendez-vous. *Turbo Training* assure d'ailleurs la vente au détail de tous les produits *Borland*.

TRUBO TRAINING est ouvert du lundi au vendredi, de 9h30 à 20h, au 78 rue de Turbigo, dans le 3e arrondissement à PARIS.

N°16

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)

QUANTITÉ	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	
	DISQUETTE UTILITY-DISK (Marcel COTTINI)	170 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	
	RELIURE-ÉCRIN	40 F	
	DISQUETTE-INDEX (N°1 à 12)	30 F	
	SIGNATURE (CHECK-LIST)	30 F	
	DISQUETTE DE TREMPLIN MICRO : * *(IIe et IIc)* ☐ N°1 ☐ N°4 ☐ N°7 ☐ N°10 ☐ N°13 ☐ N°16 ☐ N°2 ☐ N°5 ☐ N°8 ☐ N°11 ☐ N°14 ☐ N°3 ☐ N°6 ☐ N°9 ☐ N°12 ☐ N°15	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 ☐ N°13 ☐ N°14 ☐ N°15 ☐ N°16	30 F 33 F	

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

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

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

Mode de règlement choisi : ☐ Chèque ☐ Mandat-lettre ☐ Carte Bleue

VISA

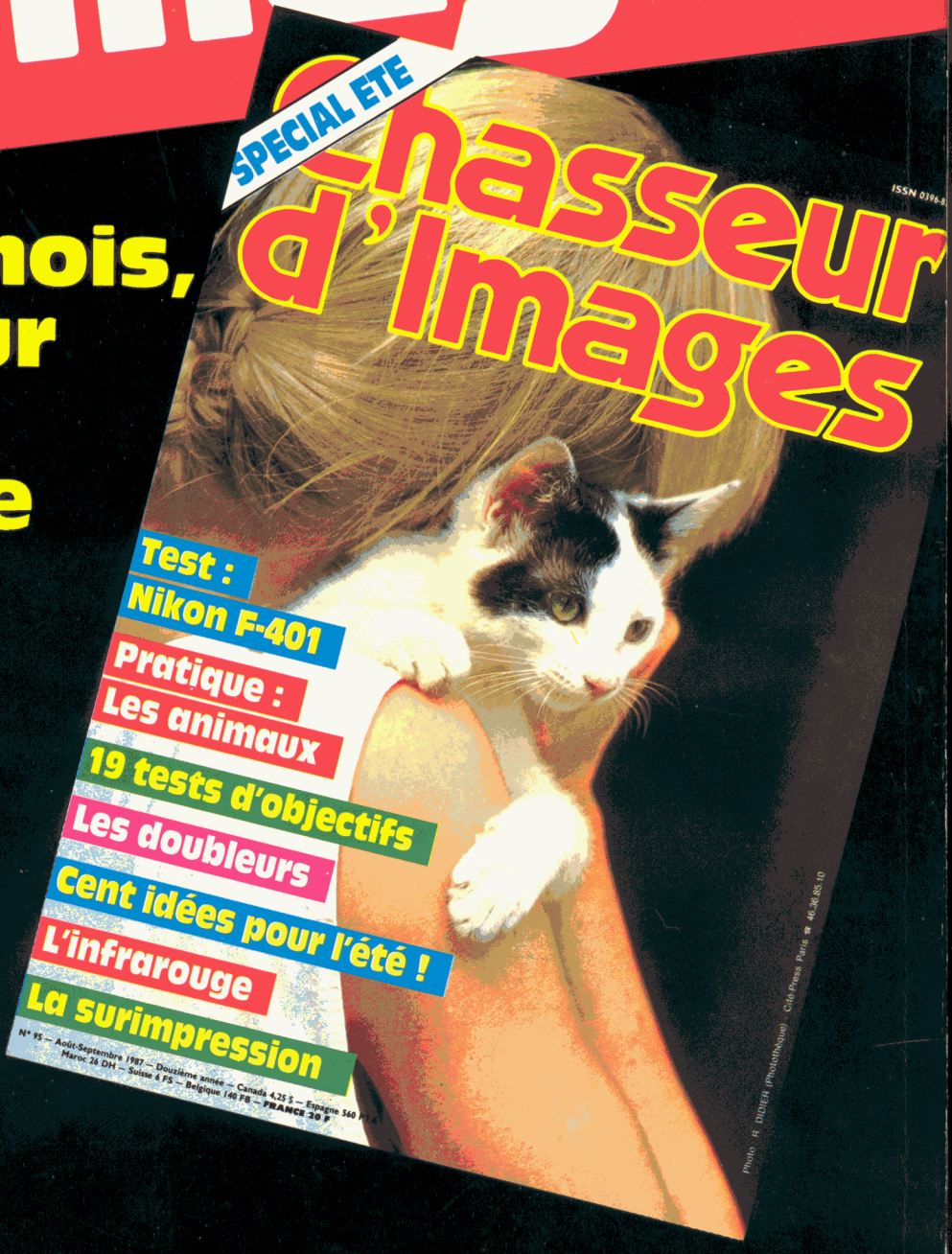
N° de votre Carte Bleue |_|_|_|_| |_|_|_|_| |_|_|_|_| |_|_|_|_|

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

Montant à débiter _____ F

Chasseur d'Images

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



Chez votre marchand de journaux !