MICRO MAG TECHNIQUE

PROGRAMMATION
BIDOUILLE
LISTING

CPC

- Stratégie, «L'omelette infernale»
- Installez votre lecteur 5"1/4

ST

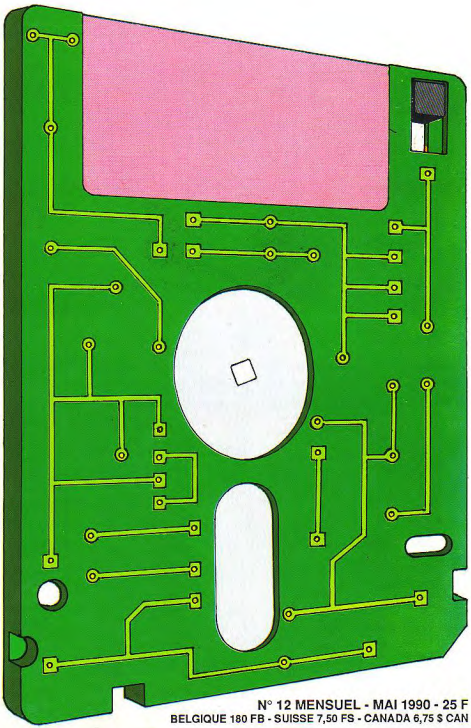
- «Paco», un jeu d'enfer

AMIGA

- Joystick automatique
- 3D en GFA

PC

- Musique et graphisme en GWBasic



M 1729 - 12 - 25,00 F

N° 12 MENSUEL - MAI 1990 - 25 F
BELGIQUE 180 FB - SUISSE 7,50 FS - CANADA 6,75 $ CAN

SOMMAIRE MAI 1990 N°12

CPC

ST

AMIGA

PC

Montage général

Puisqu'il m'échoit d'éditorialiser, rions ensemble de quelques poncifs du genre.

• **Mégalo.** ***«Depuis toujours leader incontesté de la presse informatique, le phénomène Micro-Mag crée une fois de plus l'événement avec cette nouvelle mouture qui ne laissera personne indifférent.»***

• **Démago.** ***«Ce magazine qui est le vôtre se veut représentatif de la passion qui vous anime. Quiconque peut ici exprimer son talent et affirmer ses compétences pour le plaisir de tous.»***

• **Maso.** ***«Certes imparfait, il est peu probable que ce numéro emporte l'adhésion. Toutefois, afin d'améliorer son contenu et tenter de vous satisfaire, faites-nous part de vos critiques et suggestions.»***

Mais trêve de plaisanterie. Cette revue multimachine est la seule qui souscrit à la mode diététique. Allégée de toute matière grasse ludique, elle offre en effet un assortiment complet de vitamines indispensables à votre bien-être: initiation, programmation, bidouille et listing.

Amicalement,

Jean-Claude Paulin

Directeur de la publication: Jean Kaminsky.

REDACTION
Rédacteur en chef de ce numéro:
Jean-Claude Paulin.
Ont collaboré à ce numéro: Didier Arenzana, David Farenzana, Luc et Hervé Guillaume, Michel Hugot, Claude Le Moullec, Guy Poli, Daniel Provenier, Stéphane Rodriguez, Jean-Yves Trétout.
Couverture: Claude Marrel, Jean-Claude Paulin.
Illustrations: Cécile Paulin.
Maquettistes: Jean-Claude Paulin (PAO), Jean-jacques Galmiche.

ADMINISTRATION
Abonnements: Laser Presse OGP - 175, av. Jean-Jaurès, 75019 Paris. Tél. : (1) 42 41 30 10 de 8h 30 à 18h 00 du lundi au vendredi.
Comptabilité: Sylvie Kaminsky.

REGIE PUBLICITAIRE
NEO-MEDIA, 5-7, rue de l'Amiral Courbet, 94160 Saint-Mandé. Tél. : (1) 43 98 22 22.
Chef de publicité: Pascale Kittel.

MICRO-MAG
est édité par Laser Presse SA, 5-7, rue de l'Amiral Courbet, 94160 Saint-Mandé.
Commision paritaire: n°71178.
Dépôt légal: 2e trimestre 1990.
Impression: RBI.

Premiers pas avec DEVPAC

L'ASSEMBLEUR EN DOUCEUR
(7e partie)

Ensemble nous avons tant appris, qu'il est temps d'envisager maintenant le travail sous Assembleur.

Attention, certains ordres d'installation sont relatifs à l'Assembleur DEVPAC (pour un autre Assembleur, consultez votre manuel).

- Lancez DEVPAV.
- A la question «adresse», répondez 2000.
- «*Do you want Lisa*», répondez N.

Vous êtes maintenant sous Assembleur, tapez les caractères suivants en les validant par RETURN:

- W (*Width* = largeur): permet de passer du mode 1 au mode 2 ou vice-versa. Le mode 2 est préférable car il évite, lors du listage écran, les lignes dédoublées avec commentaires. Le manque de lisibilité du mode 2 est compensé par cette propreté du listing.
- H (*Help* = au secours): donne à l'écran une liste des commandes disponibles, avec une lettre majuscule dans chacune d'elles qui devra être donnée (suivie de RETURN) pour l'exécution.
- I10,5 (*Insert*): «I» permet la numérotation automatique des lignes du programme comme AUTO en basic. Les deux valeurs qui le suivent indiquent le premier numéro et l'incrément pour les lignes suivantes.

(Les autres commandes seront détaillées lorsque le besoin s'en fera senti).

Première routine

```
10 ORG 3000 ..... ; adresse de l'implantation future
15 LD B,8 ..... ; adressage immédiat
20 LD A,B ..... ; adressage registres
25 LD (29990),A ..... ; adressage étendu
30 LD HL,29991 ..... ; adressage immédiat
35 LD (HL),15 ..... ; adressage indirect
40 LD HL,49435 ..... ; adressage immédiat
45 LD (29992),HL ..... ; adressage étendu
50 LD IX,29989 ..... ; adressage immédiat
55 LD A,(IX+1) ..... ; adressage indexé
60 LD (29994),A ..... ; adressage étendu
65 RET ..... ; fin routine - retour basic
```

- Dès lors, derrière le 10 qui apparaît, commencez la frappe des lignes de programmation en respectant bien les espaces entre les numéros de lignes et les mnémoniques, puis entre les mnémoniques et les arguments qui les suivent. Remarquez qu'une virgule sépare les deux arguments et qu'un point virgule précède le commentaire. Ce dernier est l'équivalent du REM en basic, il sera ignoré lors de la compilation. Son rôle est capital en assembleur car il permet de s'y retrouver dans un programme source écrit depuis des mois. Dans l'exemple qui suit, il n'est employé que pour rappeler les différents modes d'adressage, mais nous verrons ensuite comment faire un commentaire de façon intelligente.
- Ne tapez pas d'espace entre le dernier argument et le point virgule, car sachez qu'en listant votre programme source, tout le texte se redisposera automatiquement en tabulations. Si vous avez commis une erreur dans le genre, oubli du point-virgule devant un commentaire, vous le constaterez

immédiatement par le défaut d'alignement.

• Validez chaque ligne avec RETURN comme en Basic. A la fin de la saisie, sortez de la numérotation automatique en pressant la touche ESC.

• Listez maintenant à l'écran avec la commande «L». Celle-ci peut, si vous le désirez, admettre deux paramètres. L10,25 ne listera que les lignes portant un numéro de 10 à 25.

Première compilation

L'étape suivante est donc la compilation en vue d'obtenir le code machine. Tapez «A» (Assemblage) puis RETURN. contentez-vous pour l'instant de répondre par une simple validation aux questions posées. L'assembleur effectue alors une première passe pour vérifier la syntaxe. S'il détecte une erreur comme par exemple ORC au lieu de ORG, il affiche la ligne fautive et le type d'erreur rencontré (voir manuel). On peut continuer pour voir les autres erreurs ou arrêter par ESC et corriger.
La correction peut s'effectuer de plusieurs manières selon l'assembleur utilisé. A ce propos, DEVPAC offre un système d'édition de lignes complet, mais son maniement est plutôt long à assimiler. On peut toujours retaper la ligne complète. Personnellement, j'emploie la même procédure qu'en basic: je liste la ligne erronée, supposant la 10 en tapant L10,10, puis, en pressant sur SHIFT, j'amène avec les touches de direction un second curseur sur la ligne 10. Je récupère toute la partie correcte en la dupliquant avec COPY sans prendre les espaces avant le point-virgule du commentaire, car le buffer n'accepte pas autant de caractères qu'en basic. Une fois les corrections de syntaxe exécutées, recommencez l'opération de compilation. Si tout est OK, le code machine est produit et installé en mémoire vive à l'adresse donnée après ORG.

Première sauvegarde

Voyons maintenant comment effectuer les sauvegardes. U (*Upper*) donne le dernier numéro de ligne, soit 65 pour notre exemple. Veillez à la présence de la cassette ou disquette, tapez P10,65,PROG1 et les lignes 10 à 65 seront sauvegardées sous le nom PROG1. Si vous tapez Q10,65,PROG1, même résultat, mais avec un fichier sur disquette au format ASCII, sans intérêt pour l'instant. On peut maintenant, si l'on veut, taper V,,PROG1. Cette commande vérifie que la sauvegarde s'est bien passée en comparant le programme actuel avec celui sauvé, c'est une sécurité de plus.
Ce fichier source que l'on garde est, répétons-le, fondamental. Par prudence, aucune exécution ne devra être tentée avant de l'avoir sauvé. Sûr de nous et de la validité de notre programme, nous pourrions cependant sauvegarder les codes machine par la commande O,,PROG1-B constituant ainsi un fichier binaire du programme objet (et vérifier éventuellement si tout s'est bien passé par V,,PROG1-B). Vous avez compris que dans toutes ces commandes sans numéros de ligne, c'est leur ensemble qui est considéré et que, bien sûr, PROG1 et PROG1-B sont donnés à titre d'exemple. Ne souriez pas, j'ai connu un débutant qui utilisait toujours Save «Nomfich.bas», procédure donnée en exemple dans le manuel! Un petit truc utile quand même: le «-B» qui permet de mémoriser la nature binaire du programme (codes machine du prog. objet).
Mais testons d'abord notre programme. B (Basic) validé par RETURN nous ramène sous basic. Tapez en direct CALL 30000. Vous êtes chanceux, tout se passe bien et vous récupérez le message READY. Si la machine est plantée, réinitialisez, relancez l'assembleur, récupérez le programme source avec la commande G,,PROG1 et recherchez ce qui ne va pas (lister sur imprimante devient quasiment indispensable pour des programmes conséquents!). Si le programme semble correct, il nous reste à vérifier, grâce à la fonction PEEK, que les valeurs stockées aux adresses 29990 et suivantes sont bien là. Tapez une à une les quatre lignes suivantes :

```
? peek (29990)
? peek (29991)
? peek (29992) + (peek (29993)* 256)
? peek (29994)
```

Vous devez obtenir successivement 8, 15, 49435 et 8. Essayez de comprendre le pourquoi de ces valeurs en suivant le programme pas-à-pas et vous rappelant que:

• Une valeur entre parenthèse indique toujours une adresse. Ne vous laissez pas piéger par (HL), c'est la même chose. Remplacer mentalement HL par la valeur qu'il représente.

• Un registre simple, double ou une case-mémoire où qu'ils soient stockés, continuent à conserver leur valeur.

• Enfin, on ne répètera jamais assez qu'une valeur 16 bits stockée en mémoire vive, occupe deux adresses sous forme inversée, poids faible, puis poids fort. Dans notre exemple, c'est bien la deuxième adresse, soit 29993 qui contient le poids fort de 49435. On doit donc multiplier la valeur chargée à cette adresse par 256.

Implantation de la routine

Voilà, reste à voir la procédure permettant d'implanter en mémoire vive, hors assembleur et pour nos utilisations ultérieures sous basic, le fichier objet qui contient nos codes machine. La première chose à faire est de garantir le non-écrasement de la routine par le basic, qui place ses variables alphanumériques (chaînes) en haut de la RAM. La commande Basic MEMORY va s'en charger. La règle, souvent énoncée, est que l'on fixe MEMORY à l'adresse d'implantation moins 1. Un MEMORY 29999 préservera donc notre routine chargée en 30000. Afin de montrer que ce n'est pas toujours le cas, j'ai volontairement choisi de stocker des valeurs dans la zone d'adresse 29990 et suivantes. Il se peut en effet que l'on ait à situer «avant» la routine, des graphismes codés ou un buffer (zone d'adresses réservées pour y ranger des valeurs). Auquel cas, cet endroit se doit d'être également protégé (soit MEMORY 29989 pour l'exemple ci-dessus).
Il ne reste plus qu'à loader le fichier PROG1-B et à lancer son exécution par un CALL, ce qui nous donne :

```
10 MEMORY 29989
20 LOAD "PROG1-B",30000
30 CALL 30000
```

Fin de la 1er étape

En principe, vous devriez avoir maintenant assimilé, tout au moins dans les grandes lignes, les connaissances (vue d'ensemble de l'architecture du CPC, système binaire et hexa, modes d'adressage et commandes de base de l'assembleur) nécessaires avant d'aborder l'étude des mnémoniques. En outre, sachez qu'il convient de maîtriser aussi parfaitement que possible les commandes de l'assembleur utilisé (ici, DEVPAC) par une étude approfondie du manuel.
Prochainement, nous découvrirons l'extraordinaire facilité de programmation qui nous est offerte par les routines systèmes de l'Amstrad. Bon travail!

Guy Poli

Le fichier de REM

LES COURS DU PROFESSEUR ALI GATOR

Nul besoin d'être particulièrement clairvoyant pour affirmer que 90% des jeux informatiques exploitent grosso-modo le même canevas. A savoir, un héros (vous) chargé d'accomplir une tâche quelconque, est contrarié dans son labeur par des éléments hostiles. Sur ce principe des plus simples, voici le thème tout aussi simple du jeu que je vais insérer dans mon fichier de REM. Bob doit ramasser toutes les pommes d'un verger afin de passer au suivant. Bien sûr, des pièges à loups et deux monstres lui faciliteront la cueillette.

Lors de notre dernière rencontre, exposant une méthode personnelle de travail, je fis allusion à la pièce maîtresse de mon dispositif: un fichier de REM.

Le fichier de REM

Première chose; chargement du fichier de REM en mémoire. Que contient-il? Tout simplement la liste des sous-programmes qui vont constituer le charpente de mon jeu.
Redéfinition - Variables de base - Création décor - Tableau +1 - Dessin tableau - Routine principale - Les monstres - Vie -1 - Perdu - Gagné - Divers - Data.
Ils ont déjà leur numéro de ligne. «Variable de base» est en 500, «Routine principale» en 2000, «Les monstres» en 5000. Avec le temps je les ai tous mémorisés, d'où gain de temps appréciable lors de la mise au point. Il est bien évident qu'une fois le travail terminé, un RENUM viendra boucher les trous.

Contenu des sous-programmes

Il s'agit maintenant de les remplir, les variables, le nombre de vies, les fonctions, la déclaration des tableaux DIM, les paramètres sonores, etc.

• **Dessin de base.** Ici commence la partie création. Création du décor commun à tous les tableaux, ainsi que l'inscription des scores, du nombre de vies et du tableau en cours. Quelques RUN sont déja possibles pour juger de l'effet obtenu.

• **Tableau +1.** Le numéro de tableau en cours. Quelques RUN sont déja possibles pour juger de l'effet obtenu.

• **Redéfinition.** Ce sont les caractères redéfinis permettant de dessiner le personnage Bob, les pommes, les murs, les pièges et les monstres. Un rapide calcul doit vous permettre de connaître le nombre de caractères redéfinis qui vous seront nécessaires. Ajoutez-y 10 avant de définir la valeur du Symbol After. Ceci afin de parer à toute éventualité. Symbol after est une commande beaucoup plus complexe qu'elle n'y paraît. Nous lui consacrerons bientôt un cours entier.

• **Variable de base.** Sous-programme contenant l'initialisation de tous les paramètres de départ du jeu: Le mode, les coutableau ayant été déclaré dans les variables de base, le branchement se fera sous la forme : ON nrtab goto 100, 110, 120, etc... Les lignes 100, 110, 120... seront du type :
RESTORE 9000 : GOSUB création tableau : GOTO routine principale.

• **Création des tableaux.** Le branchement est fait. Le dessin du tableau peut commencer. Le RESTORE permet de pointer sur les data relatifs au tableau. Les éléments du décor (pommes, murs, pièges, etc.) sont dessinés à leur position définie en data. Inutile de revenir sur ce principe largement explicité. A ce stade, un RUN permet de vérifier si le tableau se dessine correctement. Pour tester celui désiré, il suffit de changer la valeur de «nrtab» en variable de base.

• **Routine principale.** Sous-programme le plus important où a lieu l'interrogation du joystick ou du clavier. En fonction de ce test, il y aura branchement à d'autres sous-programmes réalisés pour traiter chaque déplacement ou action possible. Cette partie est la plus délicate à mettre au point, car nombreux de facteurs rentrent en ligne de compte: sens de déplacement, sortie du cadre, contacts divers, nombre de pommes ramassées,...

Il est bien évident que selon l'originalité du jeu, de nombreux autres sous-programmes non compris dans ce fichier de REM type peuvent venir se greffer à cet endroit. C'est pourquoi, dans la numérotation que j'utilise, le début de la routine principale se situe en 2000 et le sous-programme suivant, les monstres, seulement à partir de 5000. Dans cette routine principale, en plus de l'interrogation du joystick, doit être inclu un branchement sur le déplacement des monstres. Le principe est donc toujours le même: interrogation du joystick, déplacement si test positif, déplacement des monstres puis bouclage. A ce stade, le problème qui se pose est l'obtention d'un déplacement régulier des monstres même si le joueur ne se déplace pas. Donc, prévoir un timing sans faille prenant en compte le temps de déplacement ou de non-déplacement du joueur. Plusieurs méthodes de mon cru vous seront expliquées dans un proche avenir.

• **Les monstres.** Après chaque interrogation du joystick, c'est ce sous-programme qui est lu. Il gère le déplacement des monstres, calcule leur position et tient compte des collisions éventuelles. Les différentes façons de déplacer un monstre feront l'objet d'une étude.

• **Vie -1.** Lorqu'un mauvais contact s'effectue entre le joueur et ses poursuivants, le nombre de vies restantes doit être décrémenté. Certains paramètres ont besoin d'être rafraîchis (temps, score, ect.). Lorsque toutes les variables nécessaires ont été réinitialisées, il faut faire un branche-

ment sur le sous-programme TABLEAU +1.

• **Perdu.** Nombre de vies tombé à zéro, le branchement se fait ici. Toutes les variantes sont les bienvenues: musique lugubre, explosion, disparition progressive de l'écran... laissez libre cours à votre imagination. Après ceci, toutes les variables sont réinitialisées pour la reprise du jeu. La solution la plus simple consiste en un RUN sur «Variables de base».

• **Gagné.** Très peu de différence avec VIE -1. Hormis le paramètre vie qui reste inchangé et l'incrémentation du numéro de tableau, les mêmes réinitialisations sont nécessaires. Un GOTO bien placé sur VIE -1 permet l'économie de nombreuses lignes.

• **Divers.** C'est comme son nom l'indique, il s'agit d'un sous-programme fourre-tout où sont stockées toutes les petites routines d'une ou deux lignes terminées par un RETURN et utilisées à tout moment par d'autres sous-programmes. Y sont inclus des tests de mise au point comme la représentation numérique des tableaux DIM employés, tests qui disparaîtront du listing final.

• **Data.** Pour une plus grande facilité de travail, mes data figurent toujours en fin de programme; ceux relatifs à la création de tableaux comme ceux nécessaires à l'interprétation d'une mélodie.

Voilà, nous avons fait le tour des sous-programmes contenus dans mon fichier de REM qui, bien sûr, ne s'adaptera pas à toute les situations. A vous de le modifier selon votre fantaisie. Mais croyez-moi, une trame, un plan ou un fil conducteur résout déjà 50% des problèmes.

RECREATION : DE-PLACER

Voici un petit jeu, comme je les aime, privilégiant la réflexion sur l'action.

Il s'agit tout simplement de remettre dans leur bonne colonne un certain nombre de dés. Le problème est que ceux-ci bougent quatre par quatre. Vous en mettez un à sa place, trois autres s'en trouvent déplacés.

Redéfinition

Nécessité de deux caractères redéfinis pour chaque dé, ainsi qu'un certain nombre pour dessiner l'abominable saurien à droite de l'écran.

Variables de base

Du très classique, si ce n'est la façon de représenter les dés par des variables alphanumériques indicées de type DE$(1). Ceci afin de permettre le rajout d'une variable numérique entre les parenthèses.

Pour la représentation des dés est utilisée une chaîne de caractères: les caractères redéfinis pour chaque dés, avec, entre les deux, le CHR$(8) et le CHR$(10) autorisant le retour en arrière et la descente d'une ligne du curseur. Dans le tableau DIM EC(6, 4) sera stockée la représentation numérique de chaque dé. Il y a 6 colonnes sur 4 lignes, soit 24 dés possibles.

Création du décor

Ligne 700 : affichage de l'emblème de votre serviteur.

Lignes 710 - 820 : tracé des lignes des divers cadres en fonction de leur point de départ et de leur longueur, le tout contenu dans une série de data.

Affichage tableau

Lignes 900 - 1030 : branchement en fonction du numéro de tableau (1 à 12). Il ne tient qu'à vous d'en rajouter (voir *Data 12 tableaux*).

Lignes 1050 - 1090 : lecture de la valeur du dé (zéro = pas de dé) et tirage aléatoire de sa colonne et de sa ligne.

Ligne 1130 : effacer un dé.

Ligne 1140 : afficher un dé dont la valeur est dans la variable D.

Ligne 1150 : afficher le cadre.

Ligne 1160 : l'effacer en le redessinant à l'encre zéro.

Routine principale

Le joystick est obligatoire pour ce jeu. Les 4 directions de base déplacent le cadre dont la position de départ est contenu dans les variables PA et PO. L'appui simultané sur Feu plus Droite ou Gauche permet de faire tourner dans le sens désiré les dés contenus dans le cadre.

Ligne 1390 : conservation de la valeur de départ des dés contenus dans le cadre.

Lignes 1400 - 1430 : permutation et affichage des dés.

Ligne 1440 : initialisation du DIM avec les nouvelles valeurs.

Lignes 1480 - 1510 : vérification si tous les dés à leur place. Chaque case du DIM est testée. Si sa valeur est égale à zéro (pas de dé) ou égale à la valeur de la colonne, la variable BON est écrémenté. Si BON=24 tous les dés sont à leur place... C'est gagné!

Fin de partie

Branchement à ce sous-programme pour deux raisons. Si BON=24, c'est gagné. Si TOR=0, le nombre de tour auquel vous aviez droit est épuisé et vous avez perdu. Suivant le cas, vous passez au tableau suivant ou vous refaite celui que vous n'avez pas réussi.

Gestion des scores

Une ligne pour effacer et écrire le nombre de tours restants, le tableau en cours et enfin votre score.

Data 12 tableaux

Une ligne de data est réservée pour chacun des douze tableaux écrits. La possibilité dépasse le million. Un tableau se compose d'une ligne de 24 chiffres allant de 0 à 6. La seule règle à respecter et de ne pas répéter un chiffre plus de 4 fois. Logique, il n'y a que 4 lignes pour 6 colonnes. Sachez aussi que moins votre ligne contient de zéro et plus le casse-tête devient difficile voire impossible. Bonnes migraines!..

Claude Le Moullec

```
10  REM : ......................... : [1823]
20  REM :                          : [419]
30  REM :       MICRO MAG et       : [696]
40  REM :         ALI  GATOR       : [1426]
50  REM :                          : [419]
60  REM :        presentent        : [1423]
70  REM :                          : [419]
80  REM :         DE-PLACER        : [473]
90  REM :                          : [419]
100 REM : ......................... : [1823]
110 REM :                          : [419]
120 REM :       REDEFINITION       : [1622]
130 REM :                          : [419]
140 REM : ......................... : [1823]
150 SYMBOL AFTER 200               [1432]
```

```
160 SYMBOL 201,0,0,0,0,1,1,0,1          [1667]
170 SYMBOL 202,14,31,123,245,226,       [2905]
255,127,103
180 SYMBOL 203,128,224,192,224,25       [3220]
2,255,255,224
190 SYMBOL 204,0,0,0,58,239,255,2       [2001]
54,188
200 SYMBOL 205,3,3,1,0,1,3,15,62        [1814]
210 SYMBOL 206,108,113,191,220,17       [2683]
9,119,127,248
220 SYMBOL 207,15,252,128,0,192,2       [2530]
24,253,195
230 SYMBOL 208,240,0,0,0,0,0,0,12       [1947]
8
240 SYMBOL 209,124,249,0,0,0,0,0,       [1555]
0
250 SYMBOL 210,231,223,0,0,0,0,0,       [2016]
0
260 SYMBOL 211,63,255,0,0,0,0,0,0       [2345]
270 SYMBOL 212,192,192,0,0,0,0,0,       [1836]
0
280 SYMBOL 213,254,188,94,47,255,       [2394]
255,126,192
290 SYMBOL 214,254,252,30,15,255,       [2281]
193,0,0
300 SYMBOL 220,0,127,127,127,127,       [2274]
127,127,127
310 SYMBOL 221,127,127,127,127,12       [2385]
7,127,127,0
320 SYMBOL 222,0,0,0,0,0,0,0,8          [2128]
330 SYMBOL 223,8,0,0,0,0,0,0,0          [1607]
340 SYMBOL 224,0,0,32,32,0,0,0,0        [2033]
350 SYMBOL 225,0,0,0,0,2,2,0,0          [1343]
360 SYMBOL 226,0,0,32,32,0,0,0,8        [1583]
370 SYMBOL 227,8,0,0,0,2,2,0,0          [2184]
380 SYMBOL 228,C,0,34,34,0,0,0,0        [1698]
390 SYMBOL 229,0,0,0,0,34,34,0,0        [1207]
400 SYMBOL 230,0,0,34,34,0,0,0,8        [1359]
410 SYMBOL 231,8,0,0,0,34,34,0,0        [1553]
420 SYMBOL 232,0,0,34,34,0,0,0,34       [1655]
430 SYMBOL 233,34,0,0,0,34,34,0,0       [1770]
440 REM :::::::::::::::::::::::::       [1823]
450 REM :                       :       [419]
460 REM :  VARIABLES DE BASE    :       [2081]
470 REM :                       :       [419]
480 REM :::::::::::::::::::::::::       [1823]
490 CALL &BBFF:MODE 0:BORDER 0:RE       [2244]
STORE 510
500 FOR t=0 TO 15:READ a:INK t,a:       [1630]
NEXT
510 DATA 0,2,14,23,26,4,15,6,16,2       [2473]
4,25,9,18,10,8,0
520 ali3$=CHR$(209)+CHR$(210)+CHR       [2257]
$(211)+CHR$(212)
530 ali2$=CHR$(205)+CHR$(206)+CHR       [2118]
$(207)+CHR$(208)
540 ali1$=CHR$(201)+CHR$(202)+CHR       [3744]
$(203)+CHR$(204)
550 DE$(1)=CHR$(222)+CHR$(8)+CHR$       [1958]
(10)+CHR$(223)
560 DE$(2)=CHR$(224)+CHR$(8)+CHR$       [1684]
(10)+CHR$(225)
570 DE$(3)=CHR$(226)+CHR$(8)+CHR$       [2392]
(10)+CHR$(227)
580 DE$(4)=CHR$(228)+CHR$(8)+CHR$       [1963]
(10)+CHR$(229)
590 DE$(5)=CHR$(230)+CHR$(8)+CHR$       [3238]
(10)+CHR$(231)
600 DE$(6)=CHR$(232)+CHR$(8)+CHR$       [1432]
(10)+CHR$(233)
610 FD$=CHR$(220)+CHR$(8)+CHR$(10       [3004]
)+CHR$(221)
620 NR$=CHR$(22)+CHR$(0):TR$=CHR$       [1272]
(22)+CHR$(1)
630 no$=CHR$(23)+CHR$(0):xo$=CHR$       [2571]
(22)+CHR$(1)
640 WINDOW #1,1,15,1,21:SC=0:TA=1       [2481]
:DIM EC(6,4)
650 REM :::::::::::::::::::::::         [1823]
660 REM :                       :       [419]
670 REM :    DESSIN DE BASE     :       [2128]
680 REM :                       :       [419]
690 REM :::::::::::::::::::::::         [1823]
700 PLOT -10,-10,9:TAG:MOVE 498,3       [6980]
52:PRINT ali1$;:MOVE 498,336:PRIN
T ali2$;:MOVE 498,320:PRINT ali3$
;:TAGOFF
710 DATA 2,22,13,2,25,13,16,2,5,1       [2398]
6,6,5,16,10,5,16,12,5,16,16,5,16,
18,5
720 DATA 16,22,5,16,24,5,16,2,5,2       [1479]
0,2,5,16,10,3,20
730 DATA 10,3,16,16,3,20,16,3,16,       [2442]
22,3,20,22,3,2,22,4,14,22,4
740 e1=5:e2=6:RESTORE 710:FOR i=1       [2495]
TO 20:GOSUB 750:NEXT:GOTO 830
750 READ X,Y,L:x1=14+(X-1)*32:y1=       [2428]
388-(Y-1)*16
760 IF i>10 THEN 800                    [1128]
770 FOR J=0 TO 2 STEP 2:PLOT x1,y       [2455]
1+j,e1:DRAW x1+(l-1)*32,y1+j:NEXT
780 FOR J=4 TO 6 STEP 2:PLOT x1,y       [4175]
1+j,e2:DRAW x1+(l-1)*32,y1+j:NEXT
790 RETURN                              [555]
800 PLOT x1+4,y1+2,e1:DRAW x1+4,y       [2465]
1+2-(l-1)*16
810 PLOT x1,y1+2,e2:DRAW x1,y1+2-       [1096]
(l-1)*16
820 RETURN                              [555]
830 PEN 7:LOCATE 17,7:PRINT "ALI"       [10301]
:LOCATE 16,13:PRINT "SCORE":LOCAT
E 16,19:PRINT"LEVEL":PLOT -10,-10
,7:TAG:MOVE 480,16:PRINT "TOURS";
:TAGOFF
840 FOR D=1 TO 6:X1=(D*2)+1:Y1=23       [2785]
:GOSUB 1130:NEXT
850 REM :::::::::::::::::::::::         [1823]
860 REM :                       :       [419]
870 REM :  AFFICHAGE TABLEAU    :       [1343]
880 REM :                       :       [419]
890 REM :::::::::::::::::::::::         [1823]
900 ERASE EC:DIM EC(6,5):ON TA GO       [5799]
TO 910,920,930,940,950,960,970,98
0,990,1000,1010,1020,1030
910 RESTORE 1810:GOSUB 1050:GOTO        [1509]
1220
920 RESTORE 1820:GOSUB 1050:GOTO        [2315]
1220
```

```
930 RESTORE 1830:GOSUB 1050:GOTO 1220 [2202]
940 RESTORE 1840:GOSUB 1050:GOTO 1220 [2717]
950 RESTORE 1850:GOSUB 1050:GOTO 1220 [1710]
960 RESTORE 1860:GOSUB 1050:GOTO 1220 [2126]
970 RESTORE 1870:GOSUB 1050:GOTO 1220 [1995]
980 RESTORE 1880:GOSUB 1050:GOTO 1220 [1752]
990 RESTORE 1890:GOSUB 1050:GOTO 1220 [3332]
1000 RESTORE 1900:GOSUB 1050:GOTO 1220 [2315]
1010 RESTORE 1910:GOSUB 1050:GOTO 1220 [1491]
1020 RESTORE 1920:GOSUB 1050:GOTO 1220 [2133]
1030 ta=1:GOTO 900 [781]
1040 REM ::: DESSIN DES DES ::: [326]
1050 CLS #1:FOR H=1 TO 24:READ D [1586]
1060 LI=INT(RND*4)+1:CO=INT(RND*6)+1 [2715]
1070 IF EC(CO,LI)<>0 THEN 1060 [1486]
1080 X1=(CO*2)+1:Y1=(LI*5)-2:GOSUB 1130 [1896]
1090 EC(CO,LI)=D:NEXT H [1233]
1100 x=48:y=380:pa=1:po=1:GOSUB 1150 [1049]
1110 TOR=60+(ta*3):GOSUB 1710:GOSUB 1720 [1903]
1120 BON=0:FIN=0:RETURN [1321]
1130 IF D=0 THEN PEN 0:LOCATE X1,Y1:PRINT CHR$(143):LOCATE X1,Y1+1:PRINT CHR$(143):RETURN [4291]
1140 PEN 5:LOCATE X1,Y1:PRINT FD$:PEN 1:LOCATE X1,Y1:PRINT TR$;DE$(D);NR$:RETURN [6474]
1150 ORIGIN x,y:PLOT 0,0,4:DRAW 128,0:DRAW 128,-140:DRAW 0,-140:DRAW 0,0:x2=x:y2=y:RETURN [4489]
1160 ORIGIN x2,y2:PLOT 0,0,0:DRAW 128,0:DRAW 128,-140:DRAW 0,-140:DRAW 0,0:SOUND 1,100,2,5:RETURN [6132]
1170 REM ::::::::::::::::::::::::: [1823]
1180 REM : : [419]
1190 REM : ROUTINE PRINCIPALE : [2225]
1200 REM : : [419]
1210 REM ::::::::::::::::::::::::: [1823]
1220 IF FIN=1 THEN 1580 [1005]
1230 IF JOY(0)=8 AND pa<5 THEN x=x+64:pa=pa+1:GOSUB 1160:GOSUB 1150 [2942]
1240 IF JOY(0)=4 AND pa>1 THEN x=x-64:pa=pa-1:GOSUB 1160:GOSUB 1150 [3023]
1250 IF JOY(0)=1 AND po>1 THEN y=y+80:po=po-1:GOSUB 1160:GOSUB 1150 [2324]
1260 IF JOY(0)=2 AND po<3 THEN y=y-80:po=po+1:GOSUB 1160:GOSUB 1150 [3773]
1270 IF JOY(0)=24 THEN GOSUB 1310 [1535]
1280 IF JOY(0)=20 THEN GOSUB 1390 [1124]
1290 GOTO 1220 [359]
1300 REM :: 1/4 TOUR A DROITE :: [1468]
1310 v1=ec(pa,po):v2=ec(pa+1,po):v3=ec(pa,po+1):v4=ec(pa+1,po+1) [3693]
1320 d=v3:x1=(pa*2)+1:y1=(po*5)-2:GOSUB 1130 [1605]
1330 d=v1:x1=((pa+1)*2)+1:y1=(po*5)-2:GOSUB 1130 [3433]
1340 d=v4:x1=(pa*2)+1:y1=((po+1)*5)-2:GOSUB 1130 [3193]
1350 d=v2:x1=((pa+1)*2)+1:y1=((po+1)*5)-2:GOSUB 1130 [3280]
1360 ec(pa,po)=v3:ec(pa+1,po)=v1:ec(pa,po+1)=v4:ec(pa+1,po+1)=v2 [4478]
1370 GOTO 1450 [385]
1380 REM :: 1/4 TOUR A GAUCHE :: [1579]
1390 v1=ec(pa,po):v2=ec(pa+1,po):v3=ec(pa,po+1):v4=ec(pa+1,po+1) [3693]
1400 d=v2:x1=(pa*2)+1:y1=(po*5)-2:GOSUB 1130 [2584]
1410 d=v4:x1=((pa+1)*2)+1:y1=(po*5)-2:GOSUB 1130 [3080]
1420 d=v1:x1=(pa*2)+1:y1=((po+1)*5)-2:GOSUB 1130 [3762]
1430 d=v3:x1=((pa+1)*2)+1:y1=((po+1)*5)-2:GOSUB 1130 [2516]
1440 ec(pa,po)=v2:ec(pa+1,po)=v4:ec(pa,po+1)=v1:ec(pa+1,po+1)=v3 [4763]
1450 FOR T=1 TO 15:SOUND 1,50-T,1,5:NEXT [2345]
1460 TOR=TOR-1:GOSUB 1710:GOSUB 1490 [1664]
1470 IF TOR=0 THEN FIN=1:RETURN ELSE RETURN [2100]
1480 REM ::: GAGNE ? ::: [729]
1490 BON=0:FOR H=1 TO 6:FOR G=4 TO 1 STEP -1 [2186]
1500 IF EC(H,G)=H OR EC(H,G)=0 THEN BON=BON+1 [1436]
1510 NEXT G,H:IF BON=24 THEN FIN=1:RETURN ELSE RETURN [1706]
1520 REM ::::::::::::::::::::::::: [1823]
1530 REM : : [419]
1540 REM : FIN DE PARTIE : [858]
1550 REM : : [419]
1560 REM ::::::::::::::::::::::::: [1823]
1570 REM ::: GAGNE ::: [721]
1580 IF BON<>24 THEN 1630 [629]
1590 CP=TOR:FOR H=CP TO 1 STEP -1 [3990]
:TOR=TOR-1:GOSUB 1710
1600 ENV 10,15,-1,1:SOUND 1,0,10,15,10,,15 [1930]
1610 SC=SC+1:GOSUB 1730:NEXT H:TA=TA+1:GOTO 900 [2486]
1620 REM ::: PERDU ::: [1027]
1630 SOUND 1,239,20,6:SOUND 1,0,20,6:SOUND 1,239,20,6:SOUND 1,319,20,6:SOUND 1,213,20,6:SOUND 1,239,40,6:SOUND 1,319,20,6 [7574]
1640 PEN 9:FOR H=1 TO 5:LOCATE 18,4:PRINT CHR$(214):FOR T=1 TO 100:NEXT T:LOCATE 18,4:PRINT CHR$(21 [9917]
```

```
3):FOR T=1 TO 100:NEXT T,H
1650 CLS #1:GOTO 900                                    [511]
1660 REM :::::::::::::::::::::::::                      [1823]
1670 REM :                                              : [419]
1680 REM :  GESTION  COMPTEURS                          : [2235]
1690 REM :                                              : [419]
1700 REM :::::::::::::::::::::::::                      [1823]
1710 PEN 0:LOCATE 17,23:PRINT CHR                       [5570]
$(143)+CHR$(143)+CHR$(143):PEN 12
:LOCATE 16,23:PRINT TR$;TOR;NR$:R
ETURN
1720 PEN 0:LOCATE 17,17:PRINT CHR                       [9587]
$(143)+CHR$(143)+CHR$(143):PEN 12
:LOCATE 17,17:PRINT TR$;TA;NR$:RE
TURN
1730 PEN 0:LOCATE 17,11:PRINT CHR                       [4111]
$(143)+CHR$(143)+CHR$(143):PEN 12
:LOCATE 16,11:PRINT TR$;SC;NR$:RE
TURN
1740 CALL &BB18:PEN 1:MODE 2:END                        [1229]
1750 PEN 5:LOCATE X1,Y1:PRINT FD$                       [6474]
:PEN 10:LOCATE X1,Y1:PRINT TR$;DE
$(D):NR$:RETURN
1760 REM :::::::::::::::::::::::::                      [1823]
1770 REM :                                              : [419]
1780 REM :   DATA 12 TABLEAUX                           : [1172]
1790 REM :                                              : [419]
1800 REM :::::::::::::::::::::::::                      [1823]
1810 DATA 1,1,1,1,2,2,2,2,0,0,0,0                       [2351]
,0,0,0,0,3,3,3,3,4,4,4,0
1820 DATA 1,1,2,2,2,2,4,4,4,6,0,0,0                     [2131]
,6,0,0,0,5,5,5,5,3,3,0,0
1830 DATA 1,1,6,6,6,6,6,5,5,3,0,3,4                     [3075]
,0,0,0,0,4,4,4,0,0,5,1,0
1840 DATA 1,1,2,2,2,2,3,3,3,4,4,4,6                     [2517]
,6,0,0,0,1,1,2,0,0,6,5,5
1850 DATA 1,1,1,1,0,0,3,3,3,3,0,0                       [2477]
,6,6,6,6,2,2,2,2,0,0,0,0
1860 DATA 1,1,2,2,4,4,3,3,5,5,6,6                       [2319]
,0,0,0,0,6,6,5,5,0,0,0,0
1870 DATA 1,2,2,3,3,3,4,4,4,4,5,5                       [2387]
,5,0,0,0,1,1,1,6,6,6,6,0
1880 DATA 1,1,2,3,3,3,4,5,5,5,5,6                       [2682]
,0,0,0,6,2,2,2,1,1,6,0,0
1890 DATA 1,1,2,2,2,2,3,3,3,3,0,0                       [2572]
,0,0,0,0,4,4,6,6,6,6,0,0
1900 DATA 1,1,1,1,3,3,0,0,0,0,0,6                       [2623]
,6,6,6,0,2,2,2,2,0,0,5,5
1910 DATA 1,1,1,1,2,2,2,2,5,5,5,5                       [2391]
,3,3,3,6,0,0,4,4,4,4,0,0
1920 DATA 1,1,1,1,2,2,2,2,3,3,3,3                       [2644]
,4,4,4,4,5,5,5,5,6,6,6,6
```

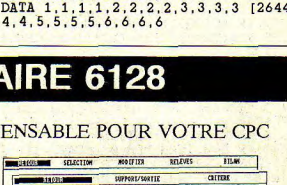
GESTION BANCAIRE 6128

LA GESTION DE COMPTE BANCAIRE INDISPENSABLE POUR VOTRE CPC

Quelques caractéristiques :

- Gère jusqu'à 10 comptes (banque, épargne, caisse...).
- Codes secrets possibles pour chacun des comptes.
- Saisie des opérations très simple, avec aide en ligne.
- Fonction archivage, vous permettant de stocker année par année vos opérations.
- Fonction TRIER, pour obtenir des listes d'opérations par dates croissantes.
- A l'aide de POINTER, vous pouvez effectuer la liaison avec votre relevé de banque 'officiel'.

- **NOMBREUSES POSSIBILITES DE SORTIES :**
  - **Recherche particulière** répondant à 1 ou plusieurs critères parmi les suivants : -DATES DE DEBUT ET DE FIN
    -MONTANTS MINIMUM ET MAXIMUM
    -UN LIBELLE PARTICULIER
  - **Liste de chèques** pouvant répondre aux critères suivants:
    -DATES DE DEBUT ET DE FIN
    -NUMEROS DE DEBUT ET DE FIN
  - **Relevé complet** -ENTRE DATES
    -AFFICHAGE SOLDE REEL OU SOLDE OPERATIONS POINTEES.
- Sorties sur ECRAN , IMPRIMANTE ou DISQUETTE.
- UTILITAIRES IMPRIMANTE, ECRAN, COPIE D'ECRAN...
- UTILISE LES 128 Ko DE VOTRE CPC 6128.



BON DE COMMANDE à retourner à

MICROLOGIC - B.P. 18 - 91211 DRAVEIL CEDEX

par téléphone:(1) 69.21.61.65 / par minitel (1) 69.24.49.08

Nom :..... Prénom:.....

Adresse:.....

Code postal:..... Ville :.....

☐ **Je commande GESTION BANCAIRE 6128 , au prix de 265,00 Frs. (port compris)**

☐ **Je désire recevoir votre CATALOGUE GRATUIT présentant votre autres produits pour CPC.**

Je choisis de régler par : Expire

☐ CHEQUE ☐ C.B. E VISA No ..... FIN .....

Signature

☐ MANDAT ☐ CONTRE RBT (+35 Frs)

MMG 12

Gare à la casse!

L'OMELETTE INFERNALE

En visite dans un poulailler particulièrement bien garni, Fox le renard espère ripailler en toute impunité. Pas si simple...

Ce jeu nécessitant réflection et stratégie utilise le joystick. Les règles sont incluses dans la présentation d'ailleurs facultative. Prenez-en connaissance aux lignes 3300 à 3340. Les inconditionnels des œuvres de Le Moullec noteront l'innovation que constitue l'accès aux écrans.

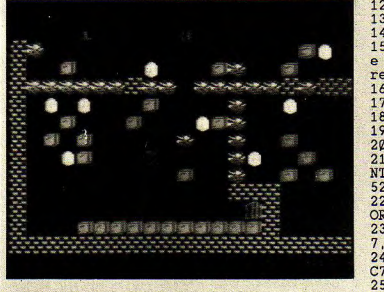
Sauvegarde

Sauvez sous un nom de votre choix, le programme Basic principal. Entrez ensuite par Amsaisie V.2 en vous reportant à son mode d'emploi, les deux listings de codes hexadécimaux.

Nom	Adr. déb.	Long
SPRITEGG	&8000	&4FF
ROUTEGG	&A000	&451

La longueur est ici précisée à l'attention de ceux qui envisagent de morceler leur travail en plusieurs fichiers qui devront ultérieurement être réunis en deux fichiers définitifs.

Claude Le Moullec



```
10 REM  ::::::::::::::::::::::::  [1823]
20 REM  :                      :  [419]
30 REM  : L'OMELETTE INFERNALE :  [1514]
40 REM  :                      :  [419]
50 REM  ::::::::::::::::::::::::  [1823]
60 REM  :                      :  [419]
70 REM  :    LMC SOFTWARE      :  [538]
80 REM  :                      :  [419]
90 REM  :  Claude  LE MOULLEC  :  [2031]
100 REM :                      :  [419]
110 REM ::::::::::::::::::::::::  [1823]
```

```
120 MEMORY &7FFF                  [150]
130 LOAD "!spritegg",&8000        [1059]
140 LOAD "!routegg",&A000         [1608]
150 GOSUB 2970:REM supprimez cett [5304]
e ligne si vous ne tapez pas la p
resentation
160 REM ::::::::::::::::::::::    [1736]
170 REM :                    :    [419]
180 REM :  VARIABLES DE BASE :    [2081]
190 REM :                    :    [419]
200 REM ::::::::::::::::::::::    [1736]
210 DEFINT a-z:CLS:LOCATE 1,1:PRI [4431]
NT "30 s DE PATIENCE....":GOSUB 2
520
220 MODE 0:BORDER 0:RESTORE 230:F [2588]
OR h=0 TO 15:READ a:INK h,a:NEXT
230 DATA 0,2,3,11,9,24,15,6,0,16, [2024]
7,13,26,24,9,18
240 DIM m1(20):DEF FN POK(A,B)=&9 [1364]
C7E+(B-1)*39+A
250 ENV 1,100,3,1:ENT 1,100,2,2   [1021]
260 ENT 2,10,-2,2:ENV 2,10,-1,2   [1423]
270 WINDOW #1,1,20,2,25:oeuf=0:tr [2116]
=1:vie=3
280 REM ::::::::::::::::::::::    [1736]
290 REM :                    :    [419]
300 REM :   BRANCHEMENT TAB  :    [1219]
310 REM :                    :    [419]
320 REM ::::::::::::::::::::::    [1736]
330 ON tr GOTO 340,350,360,370,22 [1135]
40
340 POKE &A446,&FE:POKE &A447,&98 [2019]
:CALL &A445:GOTO 410
350 POKE &A446,&7D:POKE &A447,&95 [3404]
```

```
:CALL &A445:NBO=OF:GOSUB 2810:GOT
O 400                                    
360 POKE &A446,&FC:POKE &A447,&91 [3667]
:CALL &A445:NBO=OF:GOSUB 2810:GOT
O 390                                    
370 POKE &A446,&7C:POKE &A447,&8E [4061]
:CALL &A445:NBO=OF:GOSUB 2810:GOT
O 380                                    
380 vx=32:vy=18:m1(18)=1          [1277]
390 nx=30:ny=10:m1(16)=1          [870]
400 mx=15:my=18:m1(14)=1          [1624]
410 PEN 7:LOCATE 1,1:PRINT "TAB   [4520]
 VIE    OEUF          "
420 PEN 15:LOCATE 4,1:PRINT TR:LO [1350]
CATE 10,1:PRINT vie
430 REM : ::::::::::::::::::::::  [1736]
440 REM :                      :  [419]
450 REM : ROUTINE  PRINCIPALE  :  [2225]
460 REM :                      :  [419]
470 REM : ::::::::::::::::::::::  [1736]
480 POKE FN pok(1,5),4:CALL &A0A5 [4447]
:POKE FN pok(1,5),1:POKE FN pok(2
,5),9                                    
490 CALL &A432:GOSUB 1440         [991]
500 GOSUB 610                     [923]
510 CALL &A432:GOSUB 1440         [991]
520 GOSUB 880                     [1060]
530 CALL &A432:GOSUB 1440         [991]
540 GOSUB 1140                    [823]
550 GOTO 490                      [371]
560 REM : ::::::::::::::::::::::  [1736]
570 REM :                      :  [419]
580 REM : MOUVEMENTS MONSTRES  :  [2711]
590 REM :                      :  [419]
600 REM : ::::::::::::::::::::::  [1736]
610 IF m1(14)=0 THEN RETURN       [1512]
620 m=INT(RND*2)+1:MX1=MX:MY1=MY  [1062]
630 ON SM GOTO 640,840,850,860    [1919]
640 MY=MY-1:IF PEEK(FN POK(MX,MY) [2170]
)=0 THEN 670
650 IF PEEK(FN POK(MX,MY))=9 THEN [1955]
 1790
660 SM=INT(RND*4)+1:MX=MX1:MY=MY1 [1962]
:RETURN
670 ta=PEEK(&A33D):ON ta GOTO 680 [1359]
,720,760,800
680 DEF FN PO(X,Y)=&C050+(Y-1)*16 [1576]
0+(X-1)*4
690 IF mx1<21 AND my1<13 THEN CAL [3241]
L &A00D,FN PO(MX1,MY1),&8000
700 IF mx<21 AND my<13 THEN CALL  [3707]
&A00D,FN PO(MX,MY),&8340+(&40*m)
710 POKE FN POK(MX1,MY1),0:POKE F [4196]
N POK(MX,MY),14:RETURN
720 DEF FN PO(X,Y)=&C050+(Y-1)*16 [1358]
0+(X-20)*4
730 IF mx1>19 AND my1<13 THEN CAL [1793]
L &A00D,FN PO(MX1,MY1),&8000
740 IF mx>19 AND my<13 THEN CALL  [4093]
&A00D,FN PO(MX,MY),&8340+(&40*m)
750 GOTO 710                      [431]
760 DEF FN PO(X,Y)=&C050+(Y-12)*1 [1112]
60+(X-1)*4
770 IF mx1<21 AND my1>11 THEN CAL [2711]
L &A00D,FN PO(MX1,MY1),&8000
780 IF mx<21 AND my>11 THEN CALL  [3072]
&A00D,FN PO(MX,MY),&8340+(&40*m)
790 GOTO 710                      [431]
800 DEF FN PO(X,Y)=&C050+(Y-12)*1 [1007]
60+(X-20)*4
810 IF mx1>19 AND my1>11 THEN CAL [2844]
L &A00D,FN PO(MX1,MY1),&8000
820 IF mx>19 AND my>11 THEN CALL  [3713]
&A00D,FN PO(MX,MY),&8340+(&40*m)
830 GOTO 710                      [431]
840 MX=MX+1:IF PEEK(FN POK(MX,MY) [3413]
)=0 THEN 670 ELSE 650
850 MY=MY+1:IF PEEK(FN POK(MX,MY) [2855]
)=0 THEN 670 ELSE 650
860 MX=MX-1:IF PEEK(FN POK(MX,MY) [2208]
)=0 THEN 670 ELSE 650
870 REM ::: MONSTRE 2 :::         [670]
880 IF m1(16)=0 THEN RETURN ELSE  [2365]
NX1=NX:NY1=NY
890 ON SN GOTO 900,1100,1110,1120 [2042]
900 NY=NY-1:IF PEEK(FN POK(NX,NY) [2918]
)=0 THEN 930
910 IF PEEK(FN POK(nX,nY))=9 THEN [1934]
 1790
920 SN=INT(RND*4)+1:NX=NX1:NY=NY1 [1334]
:RETURN
930 ta=PEEK(&A33D):ON ta GOTO 940 [1718]
,980,1020,1060
940 DEF FN PO(X,Y)=&C050+(Y-1)*16 [1576]
0+(X-1)*4
950 IF Nx1<21 AND Ny1<13 THEN CAL [4993]
L &A00D,FN PO(NX1,NY1),&8000
960 IF Nx<21 AND Ny<13 THEN CALL  [2898]
&A00D,FN PO(NX,NY),&83C0+(&40*m)
970 POKE FN POK(NX1,NY1),0:POKE F [2877]
N POK(NX,NY),16:RETURN
980 DEF FN PO(X,Y)=&C050+(Y-1)*16 [1358]
0+(X-20)*4
990 IF Nx1>19 AND Ny1<13 THEN CAL [3397]
L &A00D,FN PO(NX1,NY1),&8000
1000 IF Nx>19 AND Ny<13 THEN CALL [4131]
 &A00D,FN PO(NX,NY),&83C0+(&40*m)
1010 GOTO 970                     [330]
1020 DEF FN PO(X,Y)=&C050+(Y-12)* [1112]
160+(X-1)*4
1030 IF Nx1<21 AND Ny1>11 THEN CA [3485]
LL &A00D,FN PO(NX1,NY1),&8000
1040 IF Nx<21 AND Ny>11 THEN CALL [3112]
 &A00D,FN PO(NX,NY),&83C0+(&40*m)
1050 GOTO 970                     [330]
1060 DEF FN PO(X,Y)=&C050+(Y-12)* [1007]
160+(X-20)*4
1070 IF Nx1>19 AND Ny1>11 THEN CA [2384]
LL &A00D,FN PO(NX1,NY1),&8000
1080 IF Nx>19 AND Ny>11 THEN CALL [3665]
 &A00D,FN PO(NX,NY),&83C0+(&40*m)
1090 GOTO 970                     [330]
1100 NX=NX+1:IF PEEK(FN POK(NX,NY [2448]
))=0 THEN 930 ELSE 910
1110 NY=NY+1:IF PEEK(FN POK(NX,NY [3741]
))=0 THEN 930 ELSE 910
1120 NX=NX-1:IF PEEK(FN POK(NX,NY [4073]
))=0 THEN 930 ELSE 910
1130 REM :::  MONSTRE 2  :::      [670]
1140 IF m1(18)=0 THEN RETURN ELSE [3209]
```

```
 VX1=VX:VY1=VY
1150 ON SV GOTO 1160,1360,1370,13 [1509]
80
1160 VY=VY-1:IF PEEK(FN POK(VX,VY [2668]
))=0 THEN 1190
1170 IF PEEK(FN POK(VX,VY))=9 THE [2456]
N 1790
1180 SV=INT(RND*4)+1:VX=VX1:VY=VY [1493]
1:RETURN
1190 TA=PEEK(&A33D):ON TA GOTO 12 [1972]
10,1250,1280,1320
1200 DEF FN PO(X,Y)=&C050+(Y-1)*1 [1576]
60+(X-1)*4
1210 IF Vx1<21 AND Vy1<13 THEN CA [3867]
LL &A00D,FN PO(VX1,VY1),&8000
1220 IF Vx<21 AND Vy<13 THEN CALL [2871]
 &A00D,FN PO(VX,VY),&440+(&40*m)
1230 POKE FN POK(VX1,VY1),0:POKE [2875]
FN POK(VX,VY),18:RETURN
1240 DEF FN PO(X,Y)=&C050+(Y-1)*1 [1358]
60+(X-20)*4
1250 IF Vx1>19 AND Vy1<13 THEN CA [3495]
LL &A00D,FN PO(VX1,VY1),&8000
1260 IF Vx>19 AND Vy<13 THEN CALL [3520]
 &A00D,FN PO(VX,VY),&440+(&40*m)
1270 GOTO 1230                    [365]
1280 DEF FN PO(X,Y)=&C050+(Y-12)* [1112]
160+(X-1)*4
1290 IF Vx1<21 AND Vy1>11 THEN CA [3103]
LL &A00D,FN PO(VX1,VY1),&8000
1300 IF Vx<21 AND Vy>11 THEN CALL [3090]
 &A00D,FN PO(VX,VY),&440+(&40*m)
1310 GOTO 1230                    [365]
1320 DEF FN PO(X,Y)=&C050+(Y-12)* [1007]
160+(X-20)*4
1330 IF Vx1>19 AND Vy1>11 THEN CA [3540]
LL &A00D,FN PO(VX1,VY1),&8000
1340 IF Vx>19 AND Vy>11 THEN CALL [4301]
 &A00D,FN PO(VX,VY),&440+(&40*m)
1350 GOTO 1230                    [365]
1360 VX=VX+1:IF PEEK(FN POK(VX,VY [3990]
))=0 THEN 1190 ELSE 1170
1370 VY=VY+1:IF PEEK(FN POK(VX,VY [2937]
))=0 THEN 1190 ELSE 1170
1380 VX=VX-1:IF PEEK(FN POK(VX,VY [2451]
))=0 THEN 1190 ELSE 1170
1390 REM :::::::::::::::::::::    [1736]
1400 REM :                   :    [419]
1410 REM :   S/PROG  DIVERS  :    [589]
1420 REM :                   :    [419]
1430 REM :::::::::::::::::::::    [1736]
1440 a=1+PEEK(&A425):IF a>13 THEN [2558]
 1790
1450 ON a GOTO 1460,1460,1470,164 [1937]
0,1650,1670
1460 RETURN                       [555]
1470 sp=2:ram=&8080:GOSUB 1950:a= [3617]
PEEK(FN pok(ax,ay))
1480 IF a>13 THEN 1530            [730]
1490 IF a=5 THEN 1500 ELSE IF pa= [3856]
0 THEN 1510 ELSE RETURN
1500 ax1=ax:ay1=ay:sp=0:ram=&8180 [4370]
:ON ta GOSUB 2070,2110,2150,2190:
GOTO 1750
1510 ax=ax1:ay=ay1:sp=0:ram=&8000 [5181]
:ON ta GOSUB 2070,2110,2150,2190
1520 SOUND 2,50,0,15,2,2,31:RETUR [2524]
N
1530 ax1=ax:ay1=ay:sp=0:ram=&8000 [2760]
:ON ta GOSUB 2070,2110,2150,2190
1540 m1(a)=0:mons=a:SOUND 2,50,0, [2009]
15,2,2,31
1550 IF a=14 THEN AFTER 500,1 GOS [3615]
UB 1600:RETURN
1560 IF a=16 THEN AFTER 500,2 GOS [2350]
UB 1600:RETURN
1570 IF a=18 THEN AFTER 500,3 GOS [2362]
UB 1600:RETURN
1580 AFTER 500,1 GOSUB 1600:RETUR [860]
N
1590 REM ::: TEMPO MONSTRE ::::   [920]
1600 mzx=INT(RND*39)+1:mzy=INT(RN [4419]
D*23)+1:IF PEEK(FN pok(mzx,mzy))<
>0 THEN 1600
1610 IF mons=14 THEN mx=mzx:my=mz [2631]
y:m1(14)=1:RETURN
1620 IF mons=16 THEN nx=mzx:ny=mz [4394]
y:m1(16)=1:RETURN
1630 IF mons=18 THEN vx=mzx:vy=mz [3822]
y:m1(18)=1:RETURN
1640 sp=3:ram=&80C0:GOSUB 1950:IF [2995]
 pa=0 THEN 1510 ELSE RETURN
1650 IF oeuf>11 THEN 1660 ELSE RE [1207]
TURN
1660 ax=ax1:ay=ay1:sp=0:ram=&8000 [5775]
:ON ta GOSUB 2070,2110,2150,2190:
GOTO 1870
1670 sp=5:ram=&8140:GOSUB 1950:a= [2394]
PEEK(FN pok(ax,ay))
1680 IF a=1 OR a=2 THEN 1740      [1055]
1690 IF a=4 THEN oeuf=oeuf+1:LOCA [4573]
TE 17,1:PEN 15:PRINT oeuf:GOTO 17
10
1700 RETURN                       [555]
1710 ax1=ax-1:ay1=ay:sp=4:ram=&81 [1065]
00
1720 ON ta GOSUB 2070,2110,2150,2 [1129]
190
1730 FOR h=5 TO 100 STEP 5:SOUND  [2964]
1,50+h,3,5:NEXT:RETURN
1740 ax=ax1:ay=ay1:sp=0:ram=&8180 [6903]
:ON ta GOSUB 2070,2110,2150,2190
1750 FOR t=1 TO 200:NEXT t:ram=&8 [3367]
1C0:ON ta GOSUB 2070,2110,2150,21
90
1760 FOR t=1 TO 200:NEXT t:ram=&8 [3681]
000:ON ta GOSUB 2070,2110,2150,21
90
1770 SOUND 1,100,30,15,1,2,6:RETU [2821]
RN
1780 REM ::: VIE - 1 :::          [637]
1790 FOR h=0 TO 3:mu=REMAIN(h):NE [2482]
XT:vie=vie-1
1800 oeuf=0:BORDER 26:INK 0,26:IN [2070]
K 7,26
1810 OUT &BC00,2:OUT &BD49,49:SOU [3057]
ND 4,1500,50,7,0,0,10:OUT &BC00,2
:OUT &BD49,46
1820 FOR t=1 TO 4000:NEXT:BORDER  [5673]
0:POKE &A33D,1:BORDER 0:RESTORE 1
```

```
880
1830 FOR h=&A100 TO &A107:READ a$ [3812]
:POKE h,VAL("&"+a$):NEXT
1840 RESTORE 230:FOR h=0 TO 15:RE [3471]
AD a:INK h,a:NEXT:IF vie=0 THEN 2
240
1850 CALL &A445:NBO=OF:GOSUB 2810 [3878]
:ON tr GOTO 410,400,390,380
1860 REM ::: TAB + 1 ::: [571]
1870 RESTORE 1880:FOR h=&A100 TO [3789]
&A107:READ a$:POKE h,VAL("&"+a$):
NEXT
1880 DATA 02,01,02,05,1C,9D,D4,C2 [954]
1890 POKE &A33D,1:of=oeuf:oeuf=0: [2064]
tr=tr+1:GOTO 330
1900 REM :::::::::::::::::::::: [1736]
1910 REM : : [419]
1920 REM : MOUVEMENT D'OBJETS : [1886]
1930 REM : : [419]
1940 REM :::::::::::::::::::::: [1736]
1950 pa=0:ta=PEEK(&A33D):ss=PEEK( [4760]
&A100):ON ss GOTO 1960,1970,1980,
1990
1960 ax=PEEK(&A102):ay=PEEK(&A103 [1941]
)-1:GOTO 2000
1970 ax=PEEK(&A102)+1:ay=PEEK(&A1 [1819]
03):GOTO 2000
1980 ax=PEEK(&A102):ay=PEEK(&A103 [2477]
)+1:GOTO 2000
1990 ax=PEEK(&A102)-1:ay=PEEK(&A1 [2689]
03)
2000 ON ss GOTO 2010,2020,2030,20 [1290]
40
2010 ax1=ax:ay1=ay:ay=ay-1:IF PEE [2797]
K(FN pok(ax,ay))=0 THEN 2050 ELSE
 RETURN
2020 ax1=ax:ay1=ay:ax=ax+1:IF PEE [3519]
K(FN pok(ax,ay))=0 THEN 2050 ELSE
 RETURN
2030 ax1=ax:ay1=ay:ay=ay+1:IF PEE [3912]
K(FN pok(ax,ay))=0 THEN 2050 ELSE
 RETURN
2040 ax1=ax:ay1=ay:ax=ax-1:IF PEE [4100]
K(FN pok(ax,ay))=0 THEN 2050 ELSE
 RETURN
2050 ON ta GOSUB 2070,2110,2150,2 [1129]
190
2060 pa=pa+1:GOTO 2000 [1107]
2070 DEF FN PO(X,Y)=&C050+(Y-1)*1 [1576]
60+(X-1)*4
2080 IF ax1<21 AND ay1<13 THEN CA [2934]
LL &A00D,FN PO(aX1,aY1),&8000
2090 IF ax<21 AND ay<13 THEN CALL [2504]
 &A00D,FN PO(aX,aY),ram
2100 POKE FN POK(aX1,aY1),0:POKE [1639]
FN POK(aX,aY),sp:RETURN
2110 DEF FN PO(X,Y)=&C050+(Y-1)*1 [1358]
60+(X-20)*4
2120 IF ax1>19 AND ay1<13 THEN CA [3918]
LL &A00D,FN PO(aX1,aY1),&8000
2130 IF ax>19 AND ay<13 THEN CALL [3141]
 &A00D,FN PO(ax,ay),ram
2140 GOTO 2100 [359]
2150 DEF FN PO(X,Y)=&C050+(Y-12)* [1112]
160+(X-1)*4
2160 IF ax1<21 AND ay1>11 THEN CA [2852]
LL &A00D,FN PO(aX1,aY1),&8000
2170 IF ax<21 AND ay>11 THEN CALL [1692]
 &A00D,FN PO(aX,aY),ram
2180 GOTO 2100 [359]
2190 DEF FN PO(X,Y)=&C050+(Y-12)* [1007]
160+(X-20)*4
2200 IF ax1>19 AND ay1>11 THEN CA [3368]
LL &A00D,FN PO(aX1,aY1),&8000
2210 IF ax>19 AND ay>11 THEN CALL [2432]
 &A00D,FN PO(aX,aY),ram
2220 GOTO 2100 [359]
2230 X=0:FOR H=1 TO 23:FOR G=1 TO [6031]
 39:A=PEEK(&9C7F+X):PRINT USING "
#";A;:X=X+1:NEXT G:PRINT:NEXT H
2240 REM :::::::::::::::::::::: [1736]
2250 REM : : [419]
2260 REM : FIN DE PARTIE : [858]
2270 REM : : [419]
2280 REM :::::::::::::::::::::: [1736]
2290 PEN 12:WHILE INKEY$<>"":WEND [938]
2300 a$="BRAVO C'EST PAS MAL,MAIS [7130]
 ON PEUT FAIRE MIEUX ---- POUR RE
JOUER APPUYEZ SUR UNE TOUCHE ----
"
2310 B$="CE SOIR L'OMELETTE SERA [8138]
 REMPLACEE PAR UN PLAT DE NOUILLES
 A MOINS QUE VOUS VOULIEZ REJOUER
 ---- POUR CELA APPUYEZ SUR UNE T
OUCHE ----"
2320 IF TR=5 THEN C$=A$:RESTORE 2 [4621]
450:GOTO 2340
2330 C$=B$:RESTORE 2450:GOTO 2340 [886]
2340 P=1:WHILE P<>0:READ P,D [1366]
2350 SOUND 49,P/2,INT(6*d*0.8334) [1840]
,15
2360 SOUND 42,P,INT(12*d*0.8334), [2179]
15
2370 SOUND 28,P/3,INT(6*d*0.8334) [1655]
,15:WEND
2380 FOR T=1 TO 3000:NEXT:CALL &B [2028]
CA7
2390 LOCATE 1,1:PRINT SPACE$(20): [3170]
WHILE INKEY$<>"":WEND
2400 T$="":PEN 9:WHILE t$="":SOUN [2629]
D 1,RND*600+50,5,15
2410 d$=LEFT$(c$,1) [324]
2420 LOCATE 1,1:PRINT MID$(c$,1,2 [894]
0)
2430 t$=INKEY$:c$=RIGHT$(c$,LEN(c [1888]
$)-1)+d$:WEND:CLS
2440 FOR h=14 TO 18:m1(h)=0:NEXT: [889]
GOTO 270
2450 DATA 478,2,426,1,358,1,379,1 [8696]
,426,1,319,2,319,2,319,1,284,1,37
9,1,358,1,426,2,426,2,426,1,358,1
,379,1,426,1,478,1,239,1,253,1,28
4,1,319,1,358,1,379,1,426,1,478,1
,0,0
2460 DATA 426,4,426,3,426,2,426,3 [3582]
,358,4,379,2,379,3,426,2,426,3,47
8,2,426,3,0,0
2470 REM :::::::::::::::::::::: [1736]
2480 REM : : [419]
2490 REM : CREATION DES 4 TAB : [1423]
```

```
2500 REM :                    : [419]
2510 REM :::::::::::::::::::: [1736]
2520 deb=&98FD:DEF FN POK(A,B)=de [2480]
b+(B-1)*39+A
2530 FOR h=deb TO deb+896:POKE h, [1876]
0:NEXT
2540 GOSUB 2700:x=14:z=31:GOSUB 2 [2541]
730:x=7:z=14:GOSUB 2770
2550 nbc=75:nbo=35:sp=3:GOSUB 281 [3981]
0:GOSUB 2850:nbp=5:GOSUB 2890
2560 deb=&957C:DEF FN POK(A,B)=de [1976]
b+(B-1)*39+A
2570 FOR h=deb TO deb+896:POKE h, [1876]
0:NEXT
2580 GOSUB 2700:x=18:z=18:GOSUB 2 [4118]
730:x=10:z=14:GOSUB 2770
2590 nbc=70:sp=3:GOSUB 2850:nbp=5 [3651]
:GOSUB 2890:nbp=25:GOSUB 2890
2600 deb=&91FB:DEF FN POK(A,B)=de [1857]
b+(B-1)*39+A
2610 FOR h=deb TO deb+896:POKE h, [1876]
0:NEXT
2620 GOSUB 2700:x=15:z=28:GOSUB 2 [2004]
730
2630 nbc=70:sp=3:GOSUB 2850:nbp=2 [3391]
:GOSUB 2890:nbp=27:GOSUB 2890
2640 deb=&8E7B:DEF FN POK(A,B)=de [2285]
b+(B-1)*39+A
2650 FOR h=deb TO deb+896:POKE h, [1876]
0:NEXT
2660 GOSUB 2700:x=11:z=18:GOSUB 2 [2614]
770
2670 nbc=50:sp=3:GOSUB 2850:nbc=2 [3536]
5:sp=2:GOSUB 2850:nbp=2:GOSUB 289
0
2680 RETURN                   [555]
2690 REM le cadre             [690]
2700 FOR h=1 TO 39:POKE FN POK(h, [4480]
1),1:POKE FN POK(h,23),1:NEXT
2710 FOR h=2 TO 22:POKE FN POK(1, [1686]
h),1:POKE FN POK(39,h),1:NEXT:RET
URN
2720 REM lignes verticales    [1984]
2730 FOR g=1 TO 2:FOR h=2 TO 20:y [3560]
=INT(RND*5)+1
2740 IF y=1 THEN 2750 ELSE y=2 [1571]
2750 POKE FN POK(x,h),y:NEXT h:x= [2692]
z:NEXT g:RETURN
2760 REM les horizontales     [1265]
2770 FOR g=1 TO 2:FOR h=2 TO 38:y [2530]
=INT(RND*5)+1
2780 IF y=1 THEN 2790 ELSE y=2 [1213]
2790 POKE FN POK(h,x),y:NEXT h:x= [3886]
z:NEXT g:RETURN
2800 REM les oeufs            [1401]
2810 FOR h=1 TO nbo           [709]
2820 x=INT(RND*35)+3:y=INT(RND*17 [5018]
)+3:IF PEEK(FN pok(x,y))=0 THEN 2
830 ELSE 2820
2830 POKE FN POK(x,y),5:NEXT:RETU [1262]
RN
2840 REM les cartons          [1293]
2850 FOR h=1 TO nbc           [689]
2860 x=INT(RND*35)+3:y=INT(RND*17 [4227]
)+3:IF PEEK(FN pok(x,y))=0 THEN 2
870 ELSE 2860
2870 POKE FN POK(x,y),sp:NEXT:POK [1577]
E FN POK(2,5),11:RETURN
2880 REM les portes           [973]
2890 FOR h=nbp TO nbp+10:POKE FN [3044]
POK(h,22),3:NEXT
2900 FOR h=20 TO 22:POKE FN POK(n [4998]
bp+11,h),1:NEXT:POKE FN POK(nbp+1
0,20),1:POKE FN POK(nbp+10,21),4:
RETURN
2910 REM ::::::::::::::::::::: [1823]
2920 REM :                    : [419]
2930 REM :      PRESENTATION  : [1647]
2940 REM :      (facultative) : [1278]
2950 REM :                    : [419]
2960 REM ::::::::::::::::::::: [1823]
2970 MODE 0:BORDER 0:RESTORE 2980 [2210]
:FOR h=0 TO 15:READ a:INK h,a:NEX
T
2980 DATA 0,2,3,11,9,24,3,6,6,16, [1546]
6,13,26,24,9,18
2990 RESTORE 3390:ENV 1,1,15,1,3, [2924]
-1,4,12,-1,8:EVERY 25,1 GOSUB 337
0
3000 x=0:FOR h=80 TO 112 STEP 2:P [3719]
LOT 96-x,h,11:DRAW 288+x,h:x=x+2:
NEXT
3010 FOR h=80 TO 96 STEP 2:PLOT 2 [3959]
88,h:DRAW 496,208:DRAW 520,208:NE
XT
3020 FOR h=1 TO 3:CALL &A00D,&C5D [2591]
7+(h*4),&81C0
3030 CALL &A00D,&C5CB+(h*4),&8140 [2360]
:NEXT
3040 FOR h=1 TO 100:x=INT(RND*160 [3713]
)+1:y=INT(RND*20)+1
3050 PLOT 112+x,64+y,7:x=INT(RND* [2431]
160)+1:y=INT(RND*20)+1
3060 PLOT 112+x,64+y,5:NEXT:INK 7 [2224]
,24,6:INK 5,6,24
3070 PEN 0:LOCATE 4,21:PRINT CHR$ [3976]
(22)+CHR$(1)+CHR$(215)
3080 LOCATE 9,21:PRINT CHR$(214)+ [2043]
CHR$(22)+CHR$(0)
3090 SYMBOL 249,0,138,142,138,138 [2172]
,138,234,0
3100 SYMBOL 250,0,224,128,128,128 [2607]
,128,224,0
3110 SYMBOL 251,0,238,138,138,234 [1593]
,42,238,0
3120 SYMBOL 252,0,238,132,132,196 [1943]
,132,132,0
3130 SYMBOL 253,0,174,170,170,174 [1882]
,234,170,0
3140 SYMBOL 254,0,238,168,232,204 [2418]
,168,174,0
3150 lmc$=CHR$(249)+CHR$(250)+CHR [3218]
$(251)+CHR$(252)+CHR$(253)+CHR$(2
54)
3160 PLOT -10,-10,12:TAG:MOVE 40, [2362]
50:PRINT LMC$;:TAGOFF
3170 FOR H=1 TO 5:MOVE 30,H:DRAW [4848]
610,H,4:NEXT:FOR H=6 TO 20:MOVE 3
0,H:DRAW 610,H,8:NEXT
3180 FOR H=20 TO 22:MOVE 30,H:DRA [5083]
```

```
W 610,H,1:NEXT:FOR H=1 TO 8:MOVE
H,26:DRAW H,374,4:NEXT
3190 FOR H=8 TO 20:MOVE H,26:DRAW [5699]
 H,374,8:NEXT:FOR H=20 TO 22:MOVE
 H,26:DRAW H,374,1:NEXT
3200 FOR H=616 TO 620:MOVE H,26:D [6348]
RAW H,374,1:NEXT:FOR H=620 TO 632
:MOVE H,26:DRAW H,374,8:NEXT
3210 FOR H=632 TO 636:MOVE H,26:D [5526]
RAW H,374,4:NEXT:FOR H=394 TO 400
:MOVE 30,H:DRAW 610,H,4:NEXT
3220 FOR H=382 TO 392:MOVE 30,H:D [5456]
RAW 610,H,8:NEXT:FOR H=378 TO 380
:MOVE 30,H:DRAW 610,H,1:NEXT
3230 X=20:FOR T=1 TO 22:MOVE T,X: [2875]
DRAW T,20,15:X=X-1:NEXT T
3240 X=380:FOR T=1 TO 22:MOVE T,X [3296]
:DRAW T,380:X=X+1:NEXT T
3250 X=380:FOR T=618 TO 636:MOVE [3310]
T,380:DRAW 618,X:X=X+1:NEXT T
3260 X=1:FOR T=618 TO 636:MOVE T, [2075]
20:DRAW T,X:X=X+1:NEXT T
3270 PEN 3:LOCATE 3,6:PRINT "L'OM [4105]
ELETTE":PEN 10:LOCATE 7,9:PRINT "
INFERNALE"
3280 WHILE INKEY$="":WEND:INK 1,2 [3050]
6:INK 2,13:INK 3,15
3290 MODE 1:LOCATE 11,1:PEN 3:PRI [6611]
NT "L'OMELETTE INFERNALE":LOCATE
11,2:PEN 1:PRINT "==============
======"
3300 PEN 2:LOCATE 1,4:PRINT" FOX [9603]
,le renard jubile.Il vient de tro
u-ver un poulailler remplit d'oeu
fs."
3310 PEN 2:LOCATE 1,7:PRINT" Il [13920]
va falloir qu'il en fasse sortir
leplus possible.Le principal etan
t qu'a la fin des quatres tableau
x ,il en possedeune douzaine."
3320 LOCATE 1,12:PRINT" Il devra [8453]
 se mefier des gardiens de lafer
me ,des pierres qui roulent et d
e lacasse toujours possible."
3330 LOCATE 1,16:PRINT" Les cart [8919]
ons amortissent le deplacementdes
 oeufs.Les pierres permettent d'e
cra-ser les intrus."
3340 LOCATE 1,20:PRINT" Le premi [15402]
er tableau comporte 36 oeufs .Dan
s tous les suivants,vous retrouve
rez seulement les oeufs qui auron
t precedem-ment passes la porte."
3350 LOCATE 20,25:PEN 1:PRINT "<J [3814]
OYSTICK SEULEMENT>"
3360 WHILE INKEY$="":WEND:mu=REMA [3116]
IN(1):RETURN
3370 DI:IF (SQ(1) AND 7)=0 THEN E [4834]
1:RETURN ELSE READ j,k,l:IF j=-1
THEN RESTORE 3390:GOTO 3370
3380 SOUND 1,j,k,0,l:GOTO 3370      [1874]
3390 DATA 159,48,1,142,12,1,119,1 [12731]
2,1,127,12,1,142,12,1,106,24,1,10
6,24,1,106,12,1,95,12,1,127,12,1,
119,12,1,142,24,1,142,24,1,142,12
,1,119,12,1,127,12,1,142,12,1,159
,12,1,80,12,1,84,12,1,95,12,1,106
,12,1,119,12,1,127,12,1,142,12,1,
159,48,1,142,12,1
3400 DATA 119,12,1,127,12,1,142,1 [12934]
2,1,106,24,1,106,24,1,106,12,1,95
,12,1,127,12,1,119,12,1,142,24,1,
142,24,1,142,12,1,119,12,1,127,12
,1,142,12,1,159,12,1,106,12,1,142
,12,1,127,12,1,159,24,1,159,24,0,
159,48,1,142,12,1,119,12,1,127,12
,1,142,12,1,106,24,1
3410 DATA 106,24,1,106,12,1,95,12 [11733]
,1,127,12,1,119,12,1,142,24,1,142
,24,1,142,12,1,119,12,1,127,12,1,
142,12,1,159,12,1,80,12,1,84,12,1
,95,12,1,106,12,1,119,12,1,127,12
,1,142,12,1,159,48,1,142,12,1,119
,12,1,127,12,1,142,12,1,106,24,1,
106,24,1,106,12,1
3420 DATA 95,12,1,127,12,1,119,12 [6874]
,1,142,24,1,142,24,1,142,12,1,119
,12,1,127,12,1,142,12,1,159,12,1,
106,12,1,142,12,1,127,12,1,159,48
,1,-1,-1,-1
```

```
8000:00 00 00 00 00 00 00 00-00 00:80
8008:00 00 00 00 00 00 00 00-00 00:88
8010:00 00 00 00 00 00 00 00-00 00:90
8018:00 00 00 00 00 00 00 00-00 00:98
8020:00 00 00 00 00 00 00 00-00 00:A0
8028:00 00 00 00 00 00 00 00-00 00:A8
8030:00 00 00 00 00 00 00 00-00 00:B0
8038:00 00 00 00 00 00 00 00-00 00:B8
8040:44 CC 44 CC 55 CC 55 CC-55 CC:BC
8048:55 FF 55 FF 00 00 00 00-00 00:16
8050:CC 44 CC 44 CC 55 CC 55-CC 55:CE
8058:FF 55 FF 55 00 00 00 00-00 00:D2
8060:44 CC 44 CC 55 CC 55 CC-55 CC:DC
8068:55 FF 55 FF 00 00 00 00-00 00:36
8070:CC 44 CC 44 CC 55 CC 55-CC 55:EE
8078:FF 55 FF 55 00 00 00 00-00 00:F2
8080:00 15 2A 00 00 3F 9F 00-7F 00:7B
8088:15 9F 6F 2A 3F CF 6F 9F-FF 66
8090:37 6F 37 9F 3B CF 33 3F-A4 00:A4
8098:3D 67 67 9E 3D 6F CF 39-0C 00:0C
80A0:33 CF 9E 36 36 CF CF 3C-14 00:14
80A8:CF 9E CF 9E 6D CF CF 9E-AC 00:AC
80B0:6D CF 3C 9E 14 6F CF 28-4E 00:1C
80B8:00 3D 9E 00 00 14 28 00-1C 00:1C
80C0:00 00 00 00 00 00 00 00-40 00:40
80C8:00 F0 F0 F0 50 C3 C3 E1-37 00:37
80D0:F0 F0 F0 E1 E1 C3 C3 E1-C8 00:C8
80D8:E1 D2 C3 E1 E1 E1 E1 C3-E1 00:B2
80E0:E1 C3 C3 E1 E1 C3 C3 E1-E8 00:E8
80E8:E1 C3 C3 E1 F0 C3 C3 E1-3B 00:3B
80F0:E1 C3 D2 A0 F0 F0 F0 00-AD 00:AD
80F8:00 00 00 00 00 00 00 00-78 00:78
8100:00 44 F0 00 00 C8 6A A0-6F 00:6F
8108:44 6A 6A 7A 9D 6A 6A 6A-8A 00:8A
8110:95 6A 7A E0 95 6A F0 F0-5D 00:5D
8118:95 6A 6A 6A 95 6A 6A 6A-83 00:83
8120:95 6A 6A 6A 0C 6A 6A 6A-DE 00:DE
8128:84 6A 6A 6A 95 6A 6A 6A-82 00:82
8130:95 6A 6A 6A 95 6A 6A 7A-E0 00:BB
8138:95 6A F0 F0 C0 6A 6A 6A-24 00:24
8140:00 00 00 00 00 00 00 00-C1 00:C1
8148:00 44 88 00 00 CC 33 00-16 00:16
8150:00 99 33 00 44 33 33 22-97 00:97
8158:44 33 33 22 44 33 33 22-9F 00:9F
8160:44 66 33 22 44 66 33 22-3F 00:3F
8168:44 66 33 22 44 66 33 22-47 00:47
8170:44 33 99 22 44 99 33 22-4D 00:4D
8178:00 CC 33 00 00 44 88 00-7A 00:7A
8180:00 00 00 00 00 00 00 00-01 00:01
8188:00 11 88 00 00 33 66 00-BF 00:BF
8190:00 33 66 88 11 22 33 88-BF 00:BF
8198:00 22 11 22 00 11 00 01-86 00:86
81A0:11 11 22 00 11 33 00 22-51 00:51
81A8:11 22 11 88 11 00 44 22-12 00:12
81B0:11 11 00 22 00 33 33 22-93 00:93
81B8:00 33 CC 00 00 44 88 00-53 00:53
81C0:00 00 00 00 00 00 00 00-41 00:41
81C8:00 00 00 00 00 00 00 00-49 00:49
81D0:00 00 00 00 00 00 00 00-51 00:51
81D8:00 00 22 00 00 22 33 00-F0 00:F0
81E0:00 33 44 00 33 44 33 00-27 00:27
81E8:00 33 66 66 00 33 66 66-C5 00:C5
81F0:00 33 99 66 00 33 33 CC-31 00:31
81F8:00 11 66 88 00 00 CC 00-81 00:81
```

```
8200:00 14 28 3C 00 3C 3C 3C:FE
8208:00 33 3C 3C 28 91 3C 28:A6
8210:3C 3C 3C 28 14 14 3C 28:5A
8218:00 3C 3C 00 00 10 B8 00:2E
8220:02 30 B8 01 02 30 B8 00:62
8228:01 30 92 01 00 55 75 02:96
8230:00 57 FF 02 00 01 FF 02:74
8238:00 03 FF 02 01 0F 0F 02:9D
8240:00 14 28 3C 00 3C 3C 3C:5E
8248:00 F0 3C 3C 28 D0 3C 28:DA
8250:3C 3C 3C 28 14 14 3C 28:9A
8258:00 3C 3C 00 00 10 24 00:62
8260:01 30 B8 01 02 30 FC 01:85
8268:02 30 61 02 01 30 61 03:5B
8270:00 57 AB B8 02 57 AB CF:96
8278:01 FF 03 8F 07 0F 07 0B:44
8280:00 14 28 3C 00 3C 3C 3C:7E
8288:00 F0 3C 3C 28 D0 3C 28:1A
8290:3C 3C 3C 28 14 14 3C 28:DA
8298:00 30 3C 00 00 10 24 00:A2
82A0:01 30 B8 01 02 30 B8 01:E9
82A8:02 61 30 00 01 61 75 03:14
82B0:00 57 FF AB 03 8B 57 FF:33
82B8:4F CF 03 AF 07 0B 0F 0B:12
82C0:3C 14 28 00 3C 3C 3C 00:56
82C8:3C 3C 3C 28 14 14 3C 28:E2
82D0:14 3C 3C 3C 14 3C 28 28:A6
82D8:00 3C 3C 00 00 74 20 00:1E
82E0:02 74 30 01 00 74 30 01:F0
82E8:02 61 30 02 01 BA AA 00:CD
82F0:01 FF AB 01 01 FF 02 00:7F
82F8:01 FF 03 00 01 0F 0F 02:5A
8300:3C 14 28 00 3C 3C 3C 00:97
8308:3C 3C F0 00 14 3C E0 14:9B
8310:14 3C 3C 3C 14 3C 28 28:E7
8318:00 3C 3C 00 00 18 20 00:37
8320:02 74 30 02 01 FC 30 01:6F
8328:01 92 30 01 03 92 30 02:3F
8330:47 57 AB 00 CF 57 AB 01:73
8338:4F 03 FF 02 07 0B 0F 0B:3B
8340:3C 14 28 00 3C 3C 3C 00:D7
8348:3C 3C 10 A0 14 3C E0 14:2B
8350:14 3C 3C 3C 14 3C 28 28:27
8358:00 3C 3C 00 00 18 20 00:77
8360:02 74 30 02 02 74 30 01:7F
8368:00 30 92 01 03 BA 92 02:7E
8370:57 FF AB 00 FF AB 47 03:4F
8378:5F 03 03 03 03 03 03 03:FB
8380:70 03 03 70 21 30 F0 12:27
8388:21 30 70 12 12 30 70 A1:B6
8390:12 03 03 A1 12 13 23 A1:81
8398:12 21 52 A1 01 21 52 03:0A
83A0:12 21 52 A1 01 21 52 03:E8
83A8:30 30 30 A1 21 30 30 02:F4
83B0:03 30 21 02 01 12 12 A1:F8
83B8:12 A1 12 A1 12 A1 01 02:80
83C0:70 03 03 70 21 30 F0 12:67
83C8:21 30 12 12 30 70 A1 F6
83D0:12 03 03 A1 12 13 23 A1:C2
83D8:12 21 52 A1 03 21 52 02:4C
83E0:30 30 70 01 30 30 52 32
83E8:12 30 30 12 A1 21 30 12:70
83F0:01 12 30 03 12 A1 21 02:4B
83F8:12 A1 12 A1 12 A1 01 02:60
8400:01 02 57 AB 57 AB 03 57:BC
8408:AB 03 13 23 13 33 23 13:20
8410:23 03 03 13 13 07 0B 23:00
8418:AB 0F 27 17 AB 0F 27 17:B0
8420:AB 0F 27 17 AB 0F 0B 27:60
8428:FF 2B 17 17 FF BF 2B 17:C0
8430:FF AB 03 17 57 03 03 2B:A0
8438:57 57 3F 2B 01 FF 3F 02:F2
8440:57 AB 01 02 AB 03 57 AB:9E
8448:13 23 01 57 23 13 23 23:28
8450:23 03 03 13 13 07 0B 23:40
8458:AB 0F 27 17 AB 0F 27 17:F0
8460:AB 0F 0F 17 FF 07 0B 3F:A0
8468:AB 2B 17 3F AB 17 3F 3F:C0
8470:AB 03 17 3F 57 03 03 2B:18

8478:57 FF 2B 2B 01 FF 3F 02:46
8480:01 30 30 02 12 30 30 21:CF
8488:12 C3 C3 21 13 C3 C3 23:CF
8490:21 9B 33 30 30 03 83 30:DB
8498:12 03 83 21 43 83 C3 83:9F
84A0:C3 83 43 C3 C3 03 43 C3:D0
84A8:C3 C3 C3 C3 43 03 83 C3:58
84B0:43 03 03 83 43 03 83 82:18
84B8:43 C3 C3 00 00 43 83 0B:3B
84C0:01 30 30 02 12 30 30 21:0F
84C8:12 C3 C3 21 13 C3 C3 23:0F
84D0:20 33 33 30 30 03 83 30:4A
84D8:12 03 83 21 43 83 C3 83:4F
84E0:43 03 03 83 43 03 83 82:20
84E8:C3 83 43 C3 C3 C3 C3 D8
84F0:43 83 43 83 43 43 83 20
84F8:01 C3 C3 02 00 43 83 00:7B

A000:7C C6 08 67 D0 D5 11 59:61
A008:C0 19 D1 C9 00 DD 5E 00:F1
A010:DD 56 01 DD 6E 02 DD 66:1D
A018:03 06 19 C5 E5 06 04 1A:94
A020:77 13 23 10 FA E1 CD 00:C9
A028:A0 C1 10 EF C9 00 EC 9B:2F
A030:39 00 00 00 00 00 00 25:6A
A038:2A 2E AD E5 C5 2A 2E A0:E9
A040:7E FE 00 C2 4C A0 11 00:15
A048:80 C3 5F A0 C5 E5 47 21:BB
A050:00 80 C5 11 40 00 ED 5A:0E
A058:C1 10 F5 E5 21 01 00 22:44
A060:30 A0 CD 19 A0 2A 2E A0:99
A068:23 22 2E A0 2A 30 A0 23:E3
A070:23 23 23 22 30 A0 C1 10:E1
A078:C3 E1 22 2E A0 E1 22 30:8F
A080:A0 C9 CD 32 A0 06 0B C5:3A
A088:2A 30 A0 11 00 00 ED 5A:41
A090:22 30 A0 2A 2E A0 11 27:8F
A098:00 ED 5A 22 2E A0 CD 32:79
A0A0:A0 C1 10 E3 C9 7F 9C 21:51
A0A8:50 C0 22 30 A0 2A A5 A0:D0
A0B0:22 2E A0 F5 33 CD 21 59:4B
A0B8:C0 22 30 A0 01 13 00 2A:33
A0C0:A5 A0 ED 4A 22 2E A0 C3:6A
A0C8:82 A0 21 50 C0 22 30 A0:A9
A0D0:01 AD 01 2A A5 A0 ED 4A:3A
A0D8:22 2E A0 C3 82 A0 21 50:93
A0E0:C0 22 30 A0 2A 2E A0 ED:C3
A0E8:A5 A0 ED 4A 22 2E A0 C3:92
A0F0:82 A0 22 30 A0 01 C0 01:E6
A0F8:2A B6 A0 ED 4A 22 2E A0:42
A100:82 01 02 95 1C 9D D4 C2:D5
A108:E5 F3 ED 73 1F 0B FB:47
A110:C1 C9 CD 24 BB F5 CB 47:25
A118:C2 32 A1 F1 F5 CB 4F C2:4A
A120:C8 A1 F1 F5 CB 57 C2 31:49
A128:A2 F1 F5 CB 5F C2 AB A2:7C
A130:F1 C9 3E 01 32 00 A1 2A:C3
A138:04 A1 11 27 00 ED 52 E5:E2
A140:7E C3 26 A4 4A A1 E1 C3:5E
A148:30 A1 2A 04 A1 AF 77 E1:71
A150:3E 09 77 22 04 A1 3A 00:9E
A158:A1 FE 01 CA 6D A1 FE 02:AA
A160:C2 ED A0 F5 33 CA A1:B5
A168:C3 43 A2 00 83 3A 03 A1:40
A170:3D 32 03 A1 FE 0B C2 89:0D
A178:A1 3A 3D A3 FE 01 CA 89:3B
A180:A1 FE 02 CA 89 A1 C3 3E:A4
A188:00 3A 00 A1 FE 01 C2 9A:E1
A190:A1 00 00 00 00 00 00 CB
A198:A0 A1 21 40 82 22 6B A1:C9
A1A0:CD A9 A1 CD A9 A1 C3 30:5F
A1A8:A1 CD 25 A3 2A 06 A1 01:E4
A1B0:50 00 ED 42 22 06 A1 ED:9D
A1B8:5B 6B A1 CD 00 A1 CD 19:F2
A1C0:A9 CD 2F A3 CD 2F A3 C9:8C
A1C8:3E 03 32 00 A1 2A 04 A1:88
A1D0:11 27 00 ED 5A E5 C3 40:F9
A1D8:A1 3A 03 A1 3C 32 03 A1:90

A1E0:FE 0D C2 05 A2 3A 3D A3:3C
A1E8:FE 03 CA 05 A2 FE 04 CA:89
A1F0:05 A2 3A 3D A3 FE 02 C2:BD
A1F8:3E A3 3E 04 32 3D A3 CD:2C
A200:DE A9 C3 C4 A3 3E 4A 21:CA
A208:B3 A1 77 3A 01 A1 FE 01:B1
A210:C2 1C A2 21 00 83 22 6B:6E
A218:A1 C3 22 A2 21 40 82 22:92
A220:6B A1 CD A9 A1 CD A9 A1:14
A228:3E 42 21 B3 A1 77 C3 30:7F
A230:A1 3E 04 32 00 A1 3E 02:4B
A238:32 01 A1 2A 04 A1 2B E5:C8
A240:C3 A9 A1 3A 00 A1 3C 32:3B
A248:02 A1 FE 13 C2 5F A2 3A:B6
A250:3D A3 FE 01 CA 5F A2 FE:FD
A258:03 CA 5F A2 C3 3E A3 CD:56
A260:25 A3 2A 06 A1 2B E5 11:F5
A268:40 82 CD 08 A1 CD 19 A9:77
A270:CD 2F A3 CD 25 A3 E1 2B:64
A278:E5 11 00 82 CD 08 A1 CD:29
A280:19 A9 CD 2F A3 CD 25 A3:B6
A288:E1 2B E5 11 40 82 CD 08:7B
A290:A1 CD 19 A9 CD 2F A3 CD:30
A298:25 A3 E1 2B 22 06 A1 11:B1
A2A0:00 82 CD 08 A1 CD 19 A9:6C
A2A8:C3 30 A1 3E 02 32 00 A1:66
A2B0:3E 01 32 01 A1 2A 04 A1:71
A2B8:23 E5 C3 40 A1 3A 02 A1:27
A2C0:3C 32 02 A1 FE 15 CA D9:16
A2C8:A2 3A 3D A3 FE 02 CA D9:13
A2D0:A2 FE 04 CA D9 A2 C3 D9:92
A2D8:A3 CD 25 A3 2A 06 A1 23:27
A2E0:E5 11 00 83 CD 08 A1 CD:95
A2E8:19 A9 CD 2F A3 CD 25 A3:1E
A2F0:E1 23 E5 11 C0 82 CD 08:53
A2F8:A1 CD 19 A9 CD 2F A3 CD:B8
A300:25 A3 E1 23 E5 11 40 83:F4
A308:CD 08 A1 CD 19 A9 CD 2F:EF
A310:A3 CD 25 A3 E1 23 22 06:40
A318:A1 11 C0 82 CD 08 A1 CD:C6
A320:19 A9 C3 30 A1 2A 06 A1:78
A328:23 22 06 A1 3E 00 32 00:C1
A330:06 32 C5 06 6E 00 10 FD:22
A338:C1 10 F7 C1 C9 01 3A 3D:16
A340:A3 FE 01 C2 62 A3 3A 00:DF
A348:A1 FE 02 C2 57 A3 32 3D:61
A350:A3 C3 62 A3 3A 3D A3 3E:93
A358:03 32 3D A3 CD CA A0 C3:7A
A360:C4 A3 FE 02 C2 85 A3 3A:3C
A368:00 A1 FE 04 C2 7A A3 3E:62
A370:01 32 3D A3 CD A5 A0 C3:12
A378:C4 A3 3E 04 32 3D A3 CD:34
A380:A5 A0 C3 C4 A3 FE 03 C2:F5
A388:A6 A3 3A 00 A1 FE 01 C2:F5
A390:9B A3 32 3D A3 CD A5 A0:1E
A398:C3 C4 A3 3E 04 32 3D A3:6A
A3A0:CD DE A0 C3 C4 A3 3A 00:F4
A3A8:A1 FE 01 CA BD A3 3E 02:24
A3B0:32 3D A3 CD DE A0 C3 C4:DF
A3B8:A3 3E 03 32 3D A3 CD CA:39
A3C0:A0 C3 C4 A3 3A 00 A1 FE:DA
A3C8:01 C2 DA A3 01 40 06 2A:09
A3D0:06 A1 ED 4A 22 06 A1 C3:F7
A3D8:D3 A3 ED 42 22 06 A1 E5:B3
A3E0:48 00 2A 06 A1 ED 42 22:F2
A3E8:06 A1 C3 9B A4 FE 03 C2:95
A3F0:00 A4 01 40 06 2A 06 A1:2A
A3F8:ED 42 22 06 A1 C3 9B A4:AE
A400:01 40 00 2A 06 A1 ED 4A:8C
A408:22 06 A1 C3 9B A4 FE 04:C0
A410:C2 19 A4 11 C0 82 C3 1C:D9
A418:A4 11 00 82 2A 06 A1 CD:4F
A420:19 A9 C3 30 A1 00 32 25:D1
A428:A4 FE 00 CA 4A A1 E1 C3:0B
A430:39 A1 AF 32 25 A4 C3 12:91
A438:A1 21 7F 9C 11 FE 98 01:25
A440:81 03 ED BD C9 21 FE 98:57
A448:11 7F 9C 01 81 03 ED B0:65
A450:C9 00 00 00 00 00 00 00:BD
```

Stock de coke

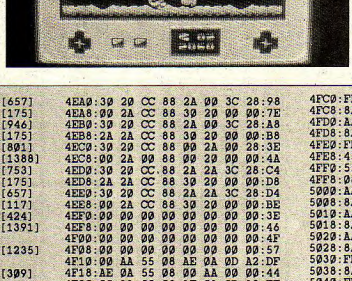
GOOD MOON

Vous disposez de neuf vies pour ramener à bord de votre navette, neuf bâtons de pixelium (espèces de stalagmites jaunes qui figurent au premier tableau). Ne pouvant en transporter qu'un à la fois, vous êtes condamné à d'incessants va-et-vient. De plus, c'est fou ce que la lune est mal fréquentée ces derniers temps. De toute façon vous n'avez pas le choix, car la navette ne décolle automatiquement qu'une fois chargée du précieux métal.

Sauvegarde

Sauvez sous le nom «GOOD-MOON» le court listing Basic de chargement. Entrez ensuite par Amsaisie V.2 en vous

Face à la pénurie de pixelium qui paralyse le fonctionnement des centrales bionucléaires, le gouvernement mondial, en vue d'une exploitation industrielle, vous charge d'aller quérir sur la lune quelques échantillons.



reportant à son mode d'emploi, les deux listings de codes hexadécimaux.

Nom	Adr. déb.	Long
GOOD1	&4E20	&12AD
GOOD2	&7530	&1CFA

La longueur est ici précisée à l'attention de ceux qui envisagent raisonnablement de morceler leur travail en plusieurs fichiers qui devront ultérieurement être réunis en deux fichiers définitifs.

Luc Guillaume
(Programmation & musique)
Hervé Guillaume
(Graphisme)

```
1 '********************  [657]
2 '*                    *  [175]
3 '* G O O D   M O O N  *  [946]
4 '*                    *  [175]
5 '*    Luc & Herve     *  [801]
6 '*    Guillaume       *  [1388]
7 '* Black System 1990  *  [753]
8 '*                    *  [175]
9 '********************  [657]
10 '                       [117]
11 MEMORY 19999            [424]
12 LOAD"!GOOD1.BIN",2000   [1391]
Ø
13 LOAD"!GOOD2.BIN",3000   [1235]
Ø
14 CALL 30000              [309]
```

```
4E20:00 00 00 00 00 00 00 00 00:6E
4E28:00 00 00 00 00 00 00 00 00:76
4E30:00 00 00 00 00 00 00 00 00:7E
4E38:00 15 00 44 00 10 00 00 00:28
4E40:55 00 04 00 45 00 51 00 00:7F
4E48:00 00 04 00 55 00 00 00 00:4B
4E50:30 20 CC 88 2A 2A 28 28 28:B8
4E58:2A 2A CC 88 30 20 00 00 00:58
4E60:30 00 CC 00 15 00 14 00 00:37
4E68:15 00 CC 88 30 20 00 00 00:FF
4E70:30 20 CC 88 00 2A 3C 28 28:92
4E78:2A 00 CC 88 30 20 00 00 00:24
4E80:30 20 CC 88 00 2A 14 28 28:8A
4E88:00 2A CC 88 30 20 00 00 00:5E
4E90:20 00 88 00 2A 2A 28 28 28:BC
4E98:3F 2A 00 88 00 20 00 00 00:59
4EA0:30 20 CC 88 2A 00 3C 28 28:98
4EA8:00 2A CC 88 30 20 00 00 00:7E
4EB0:30 20 CC 88 2A 00 3C 28 28:A8
4EB8:2A 2A CC 88 30 20 00 00 00:B8
4EC0:30 20 CC 88 00 2A 00 28 28:3E
4EC8:00 2A 00 88 00 20 00 00 00:4A
4ED0:30 20 CC 88 2A 2A 3C 28 C4:
4ED8:2A 2A CC 88 30 20 00 00 D8:
4EE0:30 20 CC 88 2A 2A 3C 28 D4:
4EE8:00 2A CC 88 30 20 00 00 00:BE
4EF0:00 00 00 00 00 00 00 00 00:3E
4EF8:00 00 00 00 00 00 00 00 00:46
4F00:00 00 00 00 00 00 00 00 00:4F
4F08:00 00 00 00 00 00 00 00 00:57
4F10:00 AA 55 08 AE 0A 0D A2:DF
4F18:AE 0A 55 08 00 AA 00 00:44
4F20:00 AA 55 08 AE 0A 0D A2:EF
4F28:AE 0A 55 08 00 AA 00 00:54
4F30:AA 00 5D 00 0E AA A7 08:53
4F38:0E AA 5D 00 AA 00 00 00:52
4F40:00 00 00 00 00 00 00 00:8F
4F48:00 00 00 00 00 00 C0 C0:D7
4F50:00 00 00 00 00 00 00 00:9F
4F58:00 00 00 00 00 00 C0 C0:E7
4F60:FF AA 0C 08 8A 8A F3 A2:E9
4F68:8A 8A 08 08 AA AA 00 00:DB
4F70:FF AA 0C 08 8A 8A F3 00:E9
4F78:8A 8A 0C 08 FF AA 00 00:A0
4F80:FF AA 0C 08 8A 00 A2 00:66
4F88:8A 00 0C 08 FF AA 00 00:9C
4F90:FF 00 0C 08 8A 8A A2 A2:8E
4F98:8A 8A 0C 08 FF 00 00 00:C4
4FA0:FF AA 0C 08 8A 00 F3 00:DD
4FA8:8A 8A 0C 08 FF 00 00 00:BC
4FB0:FF AA 08 00 8A 00 F3 00:C1
4FB8:8A 00 08 00 AA 00 00 00:FB
4FC0:FF AA 0C 08 8A 00 A2 A2:D6
4FC8:8A 8A 0C 08 FF AA 00 00:F0
4FD0:AA AA 08 08 8A 8A F3 A2:F8
4FD8:8A 8A 08 08 AA AA 00 00:4B
4FE0:FF AA 0C 08 45 00 51 00:56
4FE8:45 00 0C 08 FF AA 00 00:B7
4FF0:55 AA 04 08 00 8A 00 A2:60
4FF8:08 8A 0C 08 FF AA 00 00:9E
5000:AA AA 08 08 8A 8A F3 00:19
5008:8A 8A 08 08 AA AA 00 00:7C
5010:AA AA 00 08 00 8A 00 A2:42
5018:8A 00 0C 08 FF AA 00 00:2D
5020:AA AA AA 0C 08 8A 8A A2:1E
5028:8A 8A 08 08 AA AA 00 00:9C
5030:FF AA 0C 08 8A 8A A2 A2:83
5038:8A 8A 08 08 AA AA 00 00:AC
5040:FF AA 0C 08 8A 8A A2 A2:93
5048:8A 8A 0C 08 FF AA 00 00:71
5050:FF AA 0C 08 8A 8A F3 A2:DA
5058:8A 00 08 00 AA 00 00 00:9C
5060:FF AA 0C 08 8A 8A A2 A2:B3
5068:8A 8A 0C 08 AE AA 00 00:FC
5070:FF AA 0C 08 8A 8A A2 A2:C3
5078:CF 00 08 08 AA AA 00 00:1D
5080:FF AA 0C 08 8A 00 F3 A2:CE
5088:00 8A 0C 08 FF AA 00 00:27
5090:FF AA 0C 08 45 00 51 00:07
5098:45 00 04 00 55 00 00 00:E2
50A0:AA AA 08 08 8A 8A A2 A2:92
50A8:8A 8A 0C 08 FF AA 00 00:D1
50B0:AA AA 08 08 8A 8A A2 A2:A2
50B8:4D 08 AE AA 55 00 00 00:C0
50C0:AA AA 08 08 8A 8A A2 A2:B2
50C8:8E 8A 5D 08 AA AA 00 00:3F
50D0:AA AA 08 08 8A 8A 51 00:7B
50D8:8A 8A 08 08 AA AA 00 00:4C
```

```
5ØEØ:AA AA Ø8 Ø8 8A 8A F3 A2:Ø9
5ØE8:45 ØØ Ø4 ØØ 55 ØØ ØØ ØØ:32
5ØFØ:FF AA ØC Ø8 ØØ 8A 51 ØØ:4A
5ØF8:8A ØØ ØC Ø8 FF AA ØØ ØØ:ØD
51ØØ:1Ø 3Ø 3Ø 3Ø 3Ø CC CC CC:BD
51Ø8:64 CC 3F 3F 64 9D 3D 3F:53
511Ø:64 3E 3E 3E 64 3F 3D 3C:EC
5118:64 3E 3E 79 64 3F 3C 79:C1
512Ø:64 3E 3C F3 64 3E 79 F3:2Ø
5128:64 3E 79 F3 3Ø 3Ø 3Ø 2Ø:FØ
513Ø:CC CC CC 3Ø 3F 3F CC 98:12
5138:3F 3E 6E 98 3D 3D 3D 98:F8
514Ø:3C 3E 3F 98 B6 3D 3D 98:CD
5148:B6 3C 3F 98 F3 3C 3D 98:76
515Ø:F3 B6 3D 98 F3 B6 3D 98:85
5158:64 3E 79 F3 64 3E 3C F3:64
516Ø:64 3E 3C F3 64 3F 3C 79:EB
5168:64 3E 3E 79 64 3F 3D 3C:3Ø
517Ø:64 3E 3E 3E 64 9D 3D 3F:98
5178:64 CC 3F 3F 3Ø CC CC CC:2A
518Ø:1Ø 3Ø 3Ø 3Ø F3 B6 3D 98:FF
5188:F3 B6 3D 98 F3 3C 3D 98:E1
519Ø:B6 3C 3F 98 B6 3D 3D 98:93
5198:3C 3E 3F 98 3D 3D 3D 98:C8
51AØ:3F 3E 6E 98 3F 3F CC 98:5F
51A8:CC CC CC 3Ø 3Ø 3Ø 3Ø 2Ø:E1
51BØ:ØF 5B ØF 5B ØF 5B ØF F3:CD
51B8:F3 F3 A7 A7 A7 ØF A7 ØF:19
51CØ:A7 5B 5B 5B 5B 5B 5B 5B:29
51C8:5B 5B F3 F3 A7 ØF A7 A7:35
51DØ:A7 A7 A7 A7 5B F3 5B 59:65
51D8:59 5B 59 5B F3 F3 A7 A7:E9
51EØ:A6 A7 A6 A7 A7 A7 59 5B:27
51E8:59 59 59 59 59 59 F3 F3:C3
51FØ:A6 A7 A6 A6 A6 A6 A6 A7:A3
51F8:59 59 59 59 59 59 59 59:CD
52ØØ:F3 F3 A6 A6 A6 A6 A6 A6:91
52Ø8:A6 A6 ØC 59 ØC 59 ØC 59:42
521Ø:ØC FB F3 F3 A6 A6 A6 ØC:15
5218:A6 ØC A6 A6 FF FB FF FB:6Ø
522Ø:FF FB FF F3 F3 F3 F7 F7:1A
5228:F7 FF F7 FF F7 F7 CØ CØ:2D
523Ø:CØ DØ F3 EØ D1 F2 4A D1:8A
5238:A5 48 DØ ØE EA CØ CØ CØ:B9
524Ø:3Ø 3Ø 3Ø CC CC CC 3F 3F:57
5248:3F 3C 3C 3C F3 F3 3C 79:D2
525Ø:F3 F3 F3 B6 3F 3E F3 F3:18
5258:F3 B7 CC 9D F3 F3 F3 E6:29
526Ø:3Ø 64 F3 F3 F3 B2 CØ 9Ø:FA
5268:F3 F3 F3 E2 FF EA F3 F3:A8
527Ø:3C 68 AE EA 3C 79 3F 6A:8B
5278:AE EA 3F 7B CC C8 AE EA:B3
528Ø:CC D9 3Ø 6Ø AF EA 3Ø 71:1F
5288:CØ CØ AF EA CØ D1 D5 FF:4Ø
529Ø:FA FF FF D1 D5 ØD F1 ØC:89
5298:5D D1 D5 ØC 59 A5 5D D1:46
52AØ:D5 FF FA FF FF D1 CØ CØ:DØ
52A8:AF EA CØ D1 3Ø 6Ø AF EA:4A
52BØ:3Ø 71 CC C8 AE EA CC D9:D6
52B8:3F 6A AE EA 3F 7B 3C 68:DØ
52CØ:FF EA 3C 79 F3 E2 CØ CØ:C8
52C8:F3 F3 F3 B2 3Ø 3Ø F3 F3:E1
52DØ:3Ø CC 3F 3F 3C 3C 3F 3F:E8
52D8:CC 3Ø 64 3E F3 F3 F3 F3:28
52EØ:2Ø 1Ø F3 F3 F3 F3 3D 98:F3
52E8:3C 3F CC 3Ø 3Ø CC 3F 3C:69
52FØ:ØØ 4Ø CØ ØØ 4Ø 9Ø 9D 8Ø:ED
52F8:D1 C4 3E E2 94 94 A7 48:12
53ØØ:95 95 ØE EA C4 C4 3E E2:12
53Ø8:9Ø 9Ø 9D 68 D1 E2 CØ CØ:23
531Ø:D1 79 64 6A 84 BE B2 C8:2Ø
5318:95 89 79 4A C4 64 16 48:AB
532Ø:9Ø 9Ø 89 EA CØ 94 E2 8Ø:CC
5328:4Ø 21 79 8Ø 9Ø 98 79 8Ø:17
533Ø:C4 6Ø 16 68 9Ø 3Ø 64 6A:E5
5338:CØ CØ CØ CØ ØØ 4Ø CØ ØØ:CB
534Ø:4Ø 9Ø 9D 8Ø D1 C4 3E E2:39
5348:94 94 A7 48 95 95 ØE EA:85
535Ø:C4 C4 3E E2 9Ø 9Ø 9D 68:EC
5358:D1 E2 CØ CØ D1 79 64 6A:77
536Ø:84 BE B2 C8 95 89 79 4A:A7
5368:C4 64 16 48 9Ø 9Ø 89 EA:E8
537Ø:4Ø 34 E2 8Ø 4Ø 21 E2 ØØ:45
5378:4Ø 34 E2 8Ø 4Ø 21 3C 8Ø:C3
538Ø:4Ø 3Ø 9D 8Ø 4Ø CØ CØ 8Ø:4A
5388:ØØ 4Ø CØ ØØ 4Ø 9Ø 9D 8Ø:86
539Ø:D1 C4 3E E2 94 94 A7 48:AB
5398:95 95 ØE EA C4 C4 3E E2:AA
53AØ:9Ø 9Ø 9D 68 D1 E2 CØ CØ:BB
53A8:D1 79 64 6A 84 BE B2 C8:B8
53BØ:95 89 79 4A C4 64 16 48:43
53B8:9Ø 9Ø 89 EA CØ D1 A3 8Ø:19
53CØ:4Ø F3 29 8Ø D1 3C 12 2Ø:AF
53C8:94 Ø3 C4 6Ø 9Ø 9D 3Ø C8:8F
53DØ:CØ CØ CØ CØ ØØ CØ 8Ø ØØ:A3
53D8:4Ø 6E 6Ø 8Ø D1 3D C8 E2:72
53EØ:84 5B 68 68 D5 ØD 6A 6A:F2
53E8:D1 3D C8 C8 94 6E 6Ø 6Ø:16
53FØ:CØ CØ D1 E2 95 98 B6 E2:Ø1
53F8:C4 71 7D 48 85 B6 46 6A:9F
54ØØ:84 29 98 C8 D5 46 6Ø 6Ø:7F
54Ø8:4Ø D1 68 CØ 4Ø B6 12 8Ø:78
541Ø:4Ø B6 64 6Ø 94 29 9Ø C8:C6
5418:95 98 3Ø 6Ø CØ CØ CØ CØ:C1
542Ø:ØØ CØ 8Ø ØØ 4Ø 6E 6Ø 8Ø:E8
5428:D1 3D C8 E2 84 5B 68 68:75
543Ø:D5 ØD 6A 6A D1 3D C8 C8:94
5438:94 6E 6Ø 6Ø CØ CØ D1 E2:A3
544Ø:95 98 B6 E2 C4 71 7D 48:28
5448:85 B6 46 6A 84 29 98 C8:F9
545Ø:D5 46 6Ø 6Ø 4Ø D1 38 CØ:53
5458:ØØ D1 12 8Ø 4Ø D1 38 8Ø:32
546Ø:4Ø 3C 12 8Ø 4Ø 6E 3Ø 8Ø:C6
5468:4Ø CØ CØ 8Ø ØØ CØ 8Ø ØØ:BC
547Ø:4Ø 6E 6Ø 8Ø D1 3D C8 E2:ØB
5478:84 5B 68 68 D5 ØD 6A 6A:8B
548Ø:D1 3D C8 C8 94 6E 6Ø 6Ø:AF
5488:CØ CØ D1 E2 95 98 B6 E2:9A
549Ø:C4 71 7D 48 85 B6 46 6A:38
5498:84 29 98 C8 D5 46 6Ø 6Ø:17
54AØ:4Ø 53 E2 CØ 4Ø 16 F3 8Ø:E9
54A8:1Ø 21 3C E2 9Ø C8 Ø3 68:5F
54BØ:C4 3Ø 6E 6Ø CØ CØ CØ CØ:72
54B8:ØØ CØ CØ ØØ ØØ 4Ø 33 73:49
54CØ:8Ø ØØ 4Ø C3 93 8Ø ØØ 4Ø:3F
54C8:F3 CB E2 ØØ 4Ø CF 48 62:6D
54DØ:ØØ D1 ØC BB C2 ØØ 91 FF:8F
54D8:E3 CØ ØØ C1 F3 73 D1 8Ø:BB
54EØ:4Ø 33 92 B6 C2 4Ø C3 C4:27
54E8:35 62 ØØ CØ 35 CC E2 ØØ:34
54FØ:4Ø 9C 3A E2 ØØ 4Ø 79 68:Ø1
54F8:62 ØØ 4Ø 3E D1 C2 ØØ 4Ø:Ø7
55ØØ:9D B3 C2 ØØ 4Ø 64 63 8Ø:EB
55Ø8:ØØ ØØ 9Ø D3 8Ø ØØ ØØ 95:81
551Ø:95 8Ø ØØ ØØ C4 C4 8Ø ØØ:E6
5518:ØØ 9Ø 9Ø 6Ø ØØ 4Ø 38 B7:7D
552Ø:C8 ØØ 94 71 6E 6Ø ØØ 4Ø:23
5528:CØ CØ 8Ø ØØ CØ ØØ 4Ø 79:85
553Ø:8Ø 4Ø CØ 8Ø 14 F3 A2 CØ:49
5538:CØ CØ DB F3 F3 59 F3 B6:54
554Ø:BE 3C 29 81 79 42 4Ø CØ:F3
5548:8Ø 4Ø 79 8Ø 4Ø 79 8Ø 4Ø:9E
555Ø:3C 8Ø CØ CØ CØ 16 F3 6C:6A
5558:16 F3 38 CØ CØ CØ ØØ CØ:91
556Ø:ØØ 4Ø 79 8Ø 4Ø CØ 8Ø 14:8Ø
5568:F3 A2 CØ CØ CØ 6D F3 F3:BF
557Ø:2C F3 B6 Ø6 3C 29 D5 79:CE
5578:42 4Ø CØ 8Ø 4Ø 79 8Ø 4Ø:65
558Ø:79 8Ø 4Ø 3C 8Ø CØ CØ CØ:3E
5588:16 F3 89 16 F3 21 CØ CØ:91
559Ø:CØ ØØ CØ ØØ 4Ø 79 8Ø 4Ø:7B
5598:CØ 8Ø 14 F3 A2 CØ CØ CØ:9F
55AØ:79 DB F3 79 59 B6 16 1C:5C
55A8:29 81 FB 42 4Ø CØ 8Ø 4Ø:61
55BØ:79 8Ø 4Ø 79 8Ø 4Ø 3C 8Ø:C6
55B8:CØ CØ CØ 16 E6 29 16 B2:83
55CØ:29 CØ CØ CØ ØØ CØ ØØ 4Ø:7E
55C8:79 8Ø 4Ø CØ 8Ø 14 F3 A2:Ø3
55DØ:CØ CØ CØ 79 E7 F3 79 A6:3D
55D8:B6 16 2C 29 81 7D 42 4Ø:78
55EØ:CØ 8Ø 4Ø 79 8Ø 4Ø 79 8Ø:E8
55E8:4Ø 3C 8Ø CØ CØ CØ 16 D9:17
55FØ:29 16 71 29 CØ CØ CØ ØØ:11
55F8:CØ ØØ 4Ø 79 8Ø 4Ø CØ 8Ø:F1
56ØØ:14 F3 A2 CØ CØ CØ 79 F3:5D
56Ø8:DB 79 F3 1C 16 3C Ø9 81:91
561Ø:79 EA 4Ø CØ 8Ø 4Ø 79 8Ø:C2
5618:4Ø 79 8Ø 4Ø 3C 8Ø CØ CØ:8C
562Ø:CØ 46 F3 29 12 F3 29 CØ:6A
5628:CØ CØ ØØ CØ ØØ 4Ø 79 8Ø:8D
563Ø:4Ø CØ 8Ø 14 F3 A2 CØ CØ:E1
5638:CØ 79 F3 E7 79 F3 A6 16:FE
564Ø:3C 7D 81 79 42 4Ø CØ 8Ø:3D
5648:4Ø 79 8Ø 4Ø 79 8Ø 4Ø 3C:4D
565Ø:8Ø CØ CØ CØ 9C F3 29 34:63
5658:F3 29 CØ CØ CØ 51 5Ø A7:41
566Ø:A4 5F 59 FA A7 A4 5F ØC:1F
5668:FF 51 5B F3 B6 ØØ ØØ ØØ:CA
567Ø:CØ Ø5 1C 3C 29 ØØ ØØ ØØ:A1
5678:CØ Ø4 AB Ø3 42 FF ØØ ØØ:E7
568Ø:CØ 55 CØ CØ D5 FØ AA ØØ:EF
5688:CØ ØØ F3 F3 F2 F3 F5 ØØ:62
569Ø:CØ B2 3Ø F3 F2 F3 F2 ØØ:7Ø
5698:CØ F3 F3 71 A7 FØ A5 2Ø:97
56AØ:CØ B6 B4 3A A6 ØF ØE 88:6Ø
56A8:CØ 3D 1F 98 F7 ØC 5D 2A:EB
56BØ:CØ 6E 4C 71 F3 FF EA A2:79
56B8:CØ 98 BA F3 B6 F3 F3 28:1D
56CØ:CØ 71 F3 F3 B6 F3 F3 28:82
56C8:CØ 14 3C 3C 3D 3C 3C 2A:37
56DØ:CØ 15 3F 3F 3F 3F 3F 88:77
56D8:CØ 44 CC CC CC CC CC 2Ø:62
56EØ:CØ 1Ø 3Ø 3Ø 3Ø 3Ø 3Ø ØØ:C6
56E8:CØ 3C 3C 3C 3C 3C 3C 3C:32
56FØ:3C 3C 3C 3C 3F 3F 3F 3F:Ø4
56F8:3F 3F 3F 3F 3F 3F 3F CC:92
57ØØ:CC CC CC CC CC CC CC CC:Ø7
57Ø8:CC CC 3Ø 3Ø 3Ø 3Ø 3Ø 3Ø:F3
571Ø:3Ø 3Ø 3Ø 3Ø 3Ø 6Ø CØ CØ:B7
5718:CØ CØ CØ CØ CØ CØ CØ 9Ø:EF
572Ø:6Ø CØ CØ CØ CØ CØ CØ CØ:17
5728:CØ CØ 9Ø 6Ø CØ 64 6A CØ:ED
573Ø:CØ 3Ø 6Ø 3Ø 6Ø 9Ø 6Ø CØ:67
5738:9D 79 CØ CØ CC C8 CC C8:DE
574Ø:9Ø 6Ø CØ 79 ØE CØ CØ 2A:61
5748:6A 2A 6A 9Ø 6Ø CØ 2F 5D:6C
575Ø:CØ CØ 28 68 3C 68 9Ø 6Ø:8B
5758:CØ 9D 79 CØ CØ 2A 6A CØ:B6
576Ø:6A 9Ø 6Ø CØ 64 3E CØ CØ:Ø9
5768:CC C8 CC C8 9Ø 6Ø CØ CØ:EF
577Ø:CØ CØ CØ 3Ø 6Ø 3Ø 6Ø 9Ø:27
5778:6Ø CØ CØ CØ CØ CØ CØ CØ:6F
578Ø:CØ CØ 9Ø 6Ø CØ 3Ø 6Ø 3Ø:47
5788:6Ø 3Ø 6Ø 3Ø CØ 9Ø 6Ø CØ:3F
579Ø:CC C8 CC C8 CC C8 CC CØ:Ø7
5798:9Ø 6Ø CØ 2A 6A CØ 6A 2A:EF
57AØ:6A 95 CØ 9Ø 6Ø CØ 28 68:C3
57A8:3C 68 28 68 94 CØ 9Ø 6Ø:77
57BØ:CØ 2A 6A 6A CØ 2A 6A 95:4B
57B8:CØ 9Ø 6Ø CØ CC C8 CC C8:8F
57CØ:CC C8 CC C8 9Ø 6Ø CØ 3Ø:C7
57C8:6Ø 3Ø 6Ø 3Ø 6Ø 3Ø 6Ø 9Ø:DF
57DØ:6Ø CØ CØ CØ CØ CØ CØ CØ:C7
57D8:CØ CØ 9Ø 6Ø CØ CØ CØ CØ:1F
57EØ:CØ CØ CØ CØ CØ 9Ø 3Ø 3Ø:A7
57E8:3Ø 3Ø 3Ø 3Ø 3Ø 3Ø 3Ø 3Ø:FF
57FØ:3Ø CC CC CC CC CC CC CC:5B
57F8:CC CC CC CC 3F 3F 3F 3F:AD
58ØØ:3F 3F 3F 3F 3F 3F 3F 3C:1C
58Ø8:3C 3C 3C 3C 3C 3C 3C 3C:DØ
581Ø:3C 3C C4 CØ F3 E2 ØØ 9Ø:F3
5818:D9 F1 F1 8Ø 4Ø 71 F3 FØ:Ø9
582Ø:8Ø 4Ø F1 F2 5A C8 4Ø 5A:15
5828:A5 5A 64 CØ ØD ØE A5 9Ø:9D
583Ø:C8 8C 5D ØE CØ 64 75 AE:72
5838:48 ØØ 9Ø CØ DD 64 8Ø 4Ø:B1
584Ø:44 6Ø 9Ø C8 ØØ 1Ø 8Ø 4Ø:4C
5848:6Ø ØØ D1 F3 CØ C8 8Ø F2:BF
585Ø:F2 E6 6Ø C8 FØ F3 B2 8Ø:E6
5858:64 A5 F1 F2 8Ø 9Ø A5 5A:2C
```

```
5860:A5 80 40 5A 0D 0E C0 40:5A
5868:0D AE 4C C4 40 8C 5D BA:00
5870:98 C4 75 EE C0 60 90 C0:EF
5878:90 88 80 00 00 40 60 00:10
5880:00 D1 F3 80 00 40 F2 F2:01
5888:E2 00 40 F0 F3 F1 80 40:27
5890:A5 F1 F2 80 40 A5 5A A5:01
5898:80 40 5A 0D 0E 80 40 0D:A0
58A0:AE 5D 80 40 AE 5D EA 00:DA
58A8:00 D5 EA 80 00 00 40 80:28
58B0:00 00 00 00 00 00 00 C3:20
58B8:C0 C3 C2 C3 C2 C3 C2 33:EA
58C0:C0 33 62 33 62 33 62 D4:9A
58C8:C0 E8 E8 E8 E8 E8 E8 D4:F8
58D0:C0 E8 E8 E8 E8 E8 E8 D4:00
58D8:C0 E8 E8 E8 E8 E8 E8 33:00
58E0:62 33 62 33 62 33 62 C3:D4
58E8:C2 C3 C2 C3 C2 C3 C2 00:84
58F0:00 00 00 00 00 00 00 00:48
58F8:00 00 00 00 00 00 00 00:50
5900:00 00 00 00 00 00 00 00:59
5908:00 00 00 00 00 00 00 00:61
5910:00 00 00 00 00 00 00 00:69
5918:00 00 00 00 00 00 00 00:71
5920:00 00 00 00 00 00 00 00:79
5928:00 00 00 00 00 00 00 00:81
5930:00 00 00 00 00 00 40 C0:49
5938:80 00 00 00 91 33 E2 00:46
5940:00 00 C1 C3 73 80 80 40:A7
5948:E7 E7 93 E2 E2 D1 84 8E:F3
5950:C1 73 68 91 F7 5D 33 93:CA
5958:6A 41 73 B3 C3 63 C8 40:5B
5960:93 63 C2 C3 60 00 C1 C3:A3
5968:B6 6E C8 00 40 D1 38 A5:81
5970:60 00 00 94 3A F2 E2 00:75
5978:00 95 98 F2 68 00 00 C4:B3
5980:71 A5 6A 00 00 90 F2 0E:40
5988:C8 00 40 73 A5 5D 60 00:3C
5990:91 93 0E FB 80 00 C1 C1:85
5998:5D B7 80 00 40 40 FB 6E:49
59A0:80 00 00 40 B7 98 80 00:1C
59A8:00 85 4C 30 80 00 40 F3:87
59B0:98 64 60 00 D1 3D C8 90:04
59B8:C8 00 94 6E 60 30 60 00:ED
59C0:C0 C0 80 C0 C0 00 00 00:99
59C8:00 00 00 00 00 00 00 00:21
59D0:00 00 00 00 00 00 00 00:29
59D8:00 00 00 00 00 00 00 00:31
59E0:00 00 00 00 00 00 00 00:39
59E8:00 00 00 00 00 00 00 00:41
59F0:00 00 00 00 00 00 00 00:49
59F8:00 00 00 00 00 00 00 00:51
5A00:00 F3 51 A2 51 F3 51 F1:C1
5A08:F3 F3 F3 F1 F2 F2 F3 F1:6B
5A10:F1 F2 F1 F0 F2 F0 F0 F1:44
5A18:F0 A5 F0 5A 5A 5A 5A 0F:B0
5A20:A5 0F 0F 0F A5 0E 0F 0D:04
5A28:0C 0F 0F 0D 0C 0F 0C 0E:67
5A30:0C AE FF AE FF 0C FF D5:8B
5A38:C0 D5 C0 FF C0 00 00 00:F8
5A40:00 00 00 00 00 00 00 00:9A
5A48:00 00 00 00 00 00 00 00:A2
5A50:00 00 00 00 00 00 00 00:AA
5A58:00 00 00 00 00 00 00 00:B2
5A60:00 00 00 00 00 00 00 00:BA
5A68:00 00 00 00 00 00 00 00:C2
5A70:FF AA 00 00 00 FF 0C 5D:53
5A78:AA 00 55 0C CF 0C 5D 00:89
5A80:04 4D F3 CF 0C 00 AE DB:63
5A88:F3 F3 8E 00 4D DB F3 F3:45
5A90:E7 55 4D F3 F3 F3 E7 04:10
5A98:DB F3 F3 F3 F3 04 DB F3:C4
5AA0:F3 F3 F3 AE CF F3 F3 F3:5E
5AA8:E7 4D DB F3 F3 F3 F3 4D:5E
5AB0:F3 F3 F3 F3 F3 00 00 00:47
5AB8:00 00 00 00 00 00 00 00:12
5AC0:00 AA 00 00 00 00 00 08:AE
5AC8:00 00 00 00 00 5D 00 00:50
5AD0:00 00 00 8E 00 00 00 00:62
5AD8:00 8E 00 00 00 00 00 8E:BE
5AE0:00 00 00 00 00 8E AA 00:34
5AE8:00 00 00 0C AA 00 00 00:C4
5AF0:00 4D F3 F3 F3 F3 E7 4D:B3
5AF8:DB F3 F3 F3 E7 4D DB F3:9E
5B00:F3 F3 CF 4D DB F3 F3 F3:0B
5B08:8E 4D F3 F3 F3 E7 8E 4D:A3
5B10:E7 F3 F3 CF 4D 4D F3 F3:D9
5B18:DB CF 8E 4D DB DB E7 8E:F4
5B20:4D AE CF E7 E7 4D 0C 0C:32
5B28:4D DB CF 0C 0C 8E 08 00:EB
5B30:00 00 00 8E 08 00 00 00:EB
5B38:00 4D 08 00 00 00 00 8E:B5
5B40:08 00 00 00 00 0C 08 00:23
5B48:00 00 00 0C 08 00 00 00:FB
5B50:00 0C AA 00 00 00 00 0C:21
5B58:08 00 00 00 00 5D AA 00:8F
5B60:00 00 00 0C AA 00 00 00:3D
5B68:00 FF 4D CF 8E AE 0C AE:82
5B70:0C 8E 8E 0C AE 55 4D 4D:B4
5B78:0C 0C 0C 55 0C 8E 0C 5D:3B
5B80:5D 55 AE 0C 0C 0C AE 00:62
5B88:AE 5D 5D 5D 5D 00 AE FF:61
5B90:0C AE FF 00 55 5D FF 5D:08
5B98:FF 00 00 FF AE FF AA 00:F4
5BA0:00 00 FF AA 00 5D AA 00:74
5BA8:00 00 00 AE 00 00 00 00:BB
5BB0:00 5D 00 00 00 00 00 5D:AD
5BB8:00 00 00 00 00 AA 00 00:0F
5BC0:00 00 00 AA 00 00 00 00:C3
5BC8:00 00 00 00 00 00 00 00:23
5BD0:00 00 00 00 00 00 00 00:2B
5BD8:00 00 00 00 00 00 00 00:33
5BE0:00 00 00 00 00 00 00 00:3B
5BE8:00 00 00 00 00 00 00 00:43
5BF0:00 00 00 00 00 00 00 10:CB
5BF8:30 00 00 00 10 64 CC 00:BF
5C00:00 00 64 9D 6E 00 00 00:22
5C08:CC 3E 3D 00 00 10 9D 3C:EE
5C10:B6 00 00 44 3F 79 F3 00:E8
5C18:00 64 3E 79 B6 00 00 00:68
5C20:00 00 00 00 00 00 00 00:7C
5C28:00 00 00 00 00 00 00 00:84
5C30:00 00 00 00 00 30 00 00:AC
5C38:00 00 00 CC 20 00 00 00:64
5C40:00 6E 88 00 00 00 00 3D:F8
5C48:98 00 00 00 00 3C 6E 00:A6
5C50:00 00 00 B6 6E 20 00 00:6A
5C58:00 00 00 9D 3C B6 F3 00:3D
5C60:00 9D 79 F3 F3 00 10 9D:44
5C68:3E F3 F3 00 44 3F 3C F3:CB
5C70:F3 00 44 3E 79 F3 F3 00:37
5C78:44 3E 79 F3 F3 00 44 3F:5E
5C80:79 F3 F3 00 44 3F 79 F3:C9
5C88:B6 00 44 3F 3C F3 3C 00:E4
5C90:44 3E 79 3C 3C 3C 3D 88:86
5C98:00 00 00 F3 3D 88 00 00:21
5CA0:00 B6 3F 98 00 00 00 B6:35
5CA8:3D 98 00 00 00 F3 6C 98:D7
5CB0:00 00 00 B6 3F CC 00 00:E7
5CB8:00 B6 3F CC 00 00 00 3D:55
5CC0:6E 6E 00 00 00 3D 9D CC:7F
5CC8:00 00 00 3F CC CC 00 00:E4
5CD0:00 00 44 3E 3E 3C 3D 00:39
5CD8:44 3E 3C 3D 3D 00 44 3F:A1
5CE0:3D 3C 3F 00 10 9D 3E 3E:4E
5CE8:6E 00 44 CC 3F 3D 6E 00:59
5CF0:10 64 3F 3F 9D 00 10 CC:BE
5CF8:9D 3F CC 00 00 64 6E 6E:9D
5D00:CC 00 00 64 9D CC CC 00:26
5D08:00 30 CC CC CC 6E 6E CC:4B
5D10:00 00 00 3F CC 98 00 00:F5
5D18:00 CC 6E CC 00 00 00 6E:F7
5D20:CC 30 00 00 00 CC CC 98:C5
5D28:00 00 00 CC CC 30 00 00:D1
5D30:00 98 98 88 00 00 00 CC:05
5D38:CC 20 00 00 00 64 64 20:B5
5D40:00 00 00 98 98 00 00 00:F5
5D48:00 00 00 10 CC 64 64 00:F5
5D50:00 10 98 64 98 00 00 00:1D
5D58:64 30 64 00 00 00 10 30:95
5D60:98 00 00 00 00 10 30 00:05
5D68:00 00 00 00 00 00 00 00:C5
5D70:00 00 00 00 00 00 00 00:CD
5D78:00 00 00 00 00 00 00 00:D5
5D80:00 00 00 00 00 64 30 00:85
5D88:00 00 00 98 20 00 00 00:E5
5D90:00 30 20 00 00 00 00 30:2D
5D98:00 00 00 00 00 00 00 00:F5
5DA0:00 00 00 00 00 00 00 00:FD
5DA8:00 00 00 00 00 00 00 00:05
5DB0:00 00 00 00 00 00 00 00:0D
5DB8:00 00 00 00 00 00 00 00:15
5DC0:00 00 00 00 00 00 00 00:1D
5DC8:00 00 00 00 00 00 00 00:25
5DD0:40 C0 00 00 00 00 D1 F3:3C
5DD8:80 00 00 40 F2 F2 E2 00:49
5DE0:00 D1 F1 F2 E0 00 00 D0:5A
5DE8:F3 A5 E0 00 00 85 F0 0F:48
5DF0:E0 00 00 84 0F 58 4A 00:9E
5DF8:00 D5 0C AF 48 A2 00 51:9B
5E00:FF 0C FB F3 A2 F3 F3 FF:AB
5E08:F3 F2 F3 F0 F3 F1 F1 F0:52
5E10:F0 F2 F2 F0 F0 A5 A5 A5:11
5E18:F0 5A 5A 0F 0F 5A A5 0F:C6
5E20:0F 0E 0D 0F 0F 0D 0C 0E:69
5E28:0D 0C 0C 0F 0C 5D AE 5D:1F
5E30:FF AE FF EA D5 EA C0 D5:1B
5E38:C0 C0 80 00 00 00 00 C0:56
5E40:C0 00 00 00 00 C0 C0 00:1E
5E48:00 00 00 C0 C0 C0 00 00:E6
5E50:00 C0 C0 D1 80 00 00 C0:32
5E58:C0 D0 80 00 00 C0 C0 A7:8E
5E60:C0 00 00 C0 C0 A4 C0 00:56
5E68:00 C0 C0 5F C0 00 00 C0:C2
5E70:C0 59 C0 80 00 C0 C0 FA:10
5E78:80 00 40 C0 C0 A7 80 00:40
5E80:D1 C0 C0 A4 C0 40 D0 C0:EF
5E88:C0 5F C0 C0 A7 C0 C0 0C:07
5E90:F3 C0 A4 C0 D1 FF F2 E2:0A
5E98:5F C0 D1 F1 F2 E0 59 C0:75
5EA0:D0 F3 A5 E0 FA C0 85 F0:A8
5EA8:0F E0 A7 C0 84 0F 58 4A:70
5EB0:A4 00 00 00 00 00 00 80:B2
5EB8:00 00 00 00 00 80 00 00:16
5EC0:00 00 00 C0 00 00 00 00:1E
5EC8:00 C0 00 00 00 00 00 C0:A6
5ED0:00 00 00 00 00 C0 80 00:2E
5ED8:00 00 00 C0 C0 00 00 00:F6
5EE0:00 C0 C0 80 00 00 00 C0:FE
5EE8:D1 80 00 00 00 C0 D5 0C:CA
5EF0:AF 48 5F C0 C0 FF 0C EA:08
5EF8:0C C0 F3 E2 FF D1 FF D1:A5
5F00:F1 F1 C0 F3 F3 F2 F3 F0:CE
5F08:D1 F2 F2 F1 F2 5A D1 F1:CB
5F10:F2 5A A5 5A D0 F3 A5 0D:19
5F18:0E A5 85 F0 0F AE 5D 0E:78
5F20:84 0F 58 F7 AE 5D D5 0C:CC
5F28:AF C0 D0 80 00 00 00 C0:26
5F30:A7 80 00 00 00 C0 A4 80:32
5F38:00 00 00 E2 5F 80 00 00:FA
5F40:00 E2 59 80 00 00 00 E0:6E
5F48:FA 80 00 00 00 E0 A7 80:72
5F50:00 00 00 E0 A4 80 00 00:63
5F58:00 4A 5F 80 00 00 00 48:A8
5F60:0C 80 00 00 00 F3 F7 FB:16
5F68:F3 FF 0C F1 F3 F3 F3 F3:4E
5F70:FF F2 F3 F1 F1 F2 F1 F0:C7
5F78:F2 F0 F0 F1 F0 A5 F0 5A:2B
5F80:5A 5A 5A 0F A5 0F 0F 0F:AB
5F88:A5 0E 0F 0D 0C 0F 0F 0D:70
5F90:0C 0F 0C 0E 0C AE FF AE:2E
5F98:FF 0C FF D5 C0 D5 C0 FF:55
5FA0:C0 FB FF C0 F3 F3 D1 F3:72
5FA8:F3 F3 F2 F1 F3 F1 F2 F2:0D
5FB0:F3 F2 F3 F1 F0 F0 F0 F0:E3
5FB8:F2 5A A5 A5 A5 A5 F0 5A:B7
5FC0:0F 0F 0F 0F A5 0D 0E 0E:0E
5FC8:0D 0E 0F 0E 0D 0E 0D 0D:0D
5FD0:0C 5D FF 5D AE AE FF EA:29
5FD8:C0 EA D5 D5 C0 00 00 55:06
```

```
5FE0:00 00 00 00 00 04 00 00:57
5FE8:00 00 00 05 00 00 00 00:5B
5FF0:00 FA AA 00 00 00 AE 5B:DB
5FF8:0E AA 00 00 00 FA AA 00:3B
6000:00 00 00 05 00 00 00 00:74
6008:00 04 00 00 00 00 00 55:18
6010:00 00 00 00 00 00 00 00:70
6018:00 00 00 00 00 00 00 00:78
6020:00 00 00 00 00 00 00 20:80
6028:00 00 00 00 00 88 00 00:B8
6030:00 00 00 2A 00 00 00 00:38
6038:64 E6 20 00 00 00 00 2A:78
6040:00 00 00 00 00 88 00 00:D0
6048:00 00 00 20 00 00 00 00:28
6050:00 00 00 00 00 00 00 00:B0
6058:00 00 00 00 00 00 00 00:B8
6060:00 00 00 00 00 00 00 00:C0
6068:00 00 00 00 00 00 00 00:C8
6070:00 00 00 00 00 00 00 00:D0
6078:00 00 00 00 00 00 00 00:D8
6080:00 00 00 00 00 00 00 00:E0
6088:00 00 00 51 00 00 00 00:2C
6090:51 F3 E2 00 00 40 F3 F3:8A
6098:F3 80 00 D1 F3 F1 F3 E2:49
60A0:C0 F2 F2 F0 F1 F1 F3 F0:BA
60A8:F0 5A 5A 5A F0 A5 A5 0F:AB
60B0:0F A5 5A 0F 0F 0C 0F 0F:27
60B8:0D 0E 0E 0C 0E 0C 0E 0D:F3
60C0:0E FF 0C FF 5D FF 5D C0:A2
60C8:FF C0 EA C0 EA 00 00 00:F7
```

```
7530:C3 37 75 2A 92 CE 76 21:CD
7538:EE BD 36 C9 06 00 0E 00:5B
7540:CD 38 BC CD D8 84 AF CD:DB
7548:0E BC CD 76 76 21 64 00:52
7550:22 7E 87 3E 01 11 23 8A:20
7558:CD 19 8E 3E 0D CD 96 BB:6F
7560:CD D8 84 AF CD 0E BC CD:7B
7568:6C BB CD 76 85 DD 21 59:74
7570:88 06 0E CD D6 77 21 EB:0E
7578:87 06 06 CD A5 77 21 D0:30
7580:52 11 AC E8 06 36 0E 01:D9
7588:3E 05 CD C5 77 21 D5 52:3C
7590:11 EC CE 06 36 0E 01 3E:C9
7598:05 CD C5 77 21 DA 52 11:5E
75A0:98 C1 06 68 0E 07 3E 01:0B
75A8:CD B1 77 21 E1 52 11 CF:75
75B0:C1 06 68 0E 07 3E 01 CD:68
75B8:B1 77 21 DA 52 11 48 C1:97
75C0:06 8B 0E 02 3E 01 CD B1:E2
75C8:77 21 E6 52 11 84 C1 06:D4
75D0:8B 0E 02 3E 01 CD B1 77:4C
75D8:21 E8 52 11 4E E9 06 32:2E
75E0:0E 01 3E 04 CD C5 77 21:17
75E8:EC 52 11 5E ED 06 32 0E:2B
75F0:01 3E 04 CD C5 77 CD E9:A8
75F8:84 C3 89 77 49 4E 53 45:9C
7600:52 54 20 43 4F 49 4E 2A:8F
7608:3A BE 87 CD 1E BB C2 08:83
7610:76 06 96 CD 59 85 3E 01:93
7618:32 79 87 CD C9 7F CD E1:05
7620:7F CD D8 84 CD 76 85 CD:17
7628:68 85 21 E9 56 11 C7 E5:C4
7630:01 0B 00 3E 1B CD 39 84:B9
7638:3E 09 32 81 87 C6 30 32:BF
7640:70 89 21 70 89 11 1F D6:F7
7648:CD 7F 84 21 00 00 22 7E:77
7650:87 AF 32 83 87 32 2E 87:96
7658:32 80 87 32 B7 86 32 C8:B2
7660:86 32 D9 86 32 82 87 32:AA
7668:91 87 32 90 87 32 95 87:6D
7670:32 85 87 C3 8F 76 21 99:01
7678:87 22 98 87 21 A3 87 22:D5
7680:A2 87 21 A9 87 22 A8 87:EC
7688:21 AB 87 22 AA 87 C9 AF:05
7690:32 80 87 CD E8 83 CD 48:76
7698:85 CD 6E 82 CD 0B 85 CD:CD
76A0:E6 78 CD E9 84 C3 FB 77:B2
76A8:21 C4 76 11 BA 87 01 05:68
76B0:00 ED B0 C3 08 76 21 C9:37
76B8:76 11 BA 87 01 05 00 ED:9B
76C0:B0 C3 08 76 48 49 4A 4B:D8
76C8:4C 00 02 08 01 2F 3A 81:6D
76D0:87 3D FE 00 C2 DD 76 3E:63
76D8:01 32 80 87 AF 32 81 87:A5
76E0:21 10 00 CD 7F 86 01 50:F1
76E8:4E 09 11 1F D6 01 02 00:AF
76F0:3E 08 C3 39 84 3E FF 32:72
76F8:95 86 CD 95 7D 06 1E C5:59
7700:3E 12 CD 1E BB C4 FE 84:09
7708:CD 26 84 CD 95 7D C1 10:F6
7710:EE 06 1E C5 3E 12 CD 1E:1C
7718:BB C4 FE 84 DD 21 1D 87:F6
7720:DD 35 0C CD 26 84 CD 95:4F
7728:7D C1 10 E7 06 0A C5 3E:17
7730:12 CD 1E BB C4 FE 84 DD:E5
7738:21 1D 87 DD 34 0B CD 26:24
7740:84 CD 95 7D C1 10 E7 06:2E
7748:FA CD 59 85 CD 76 85 21:E2
7750:10 27 11 9F C9 01 30 00:17
7758:3E 64 CD 39 84 21 BF 87:EB
7760:06 02 CD A5 77 06 51 C5:B2
7768:CD 87 85 CD 6E 82 06 19:A1
7770:CD 59 85 C1 10 F1 06 FA:E9
7778:CD 59 85 CD 9C 77 CD 06:D2
7780:BB 06 96 CD 59 85 C3 60:E4
7788:75 3E 4C CD 1E BB C2 A8:8E
7790:76 3E 2F CD 1E BB C2 B6:B0
7798:76 C3 89 77 CD 00 BB 21:A8
77A0:EE BD 36 C9 C9 C5 5E 23:7A
77A8:56 23 CD 7F 84 C1 10 F5:50
77B0:C9 C5 F5 E5 D5 06 00 CD:A2
77B8:39 84 E1 CD 26 BC EB E1:E2
77C0:F1 C1 10 ED C9 C5 F5 E5:F4
77C8:D5 06 00 CD 39 84 D1 13:D8
77D0:E1 F1 C1 10 F0 C9 C5 DD:3E
77D8:E5 DD 5E 00 DD 56 01 DD:4C
77E0:6E 02 DD 66 03 DD 4E 04:77
77E8:06 00 DD 7E 06 CD 39 84:8F
77F0:DD E1 11 08 00 DD 19 C1:3E
77F8:10 DC C9 3A A3 86 FE 00:BF
7800:C4 B7 82 3A 90 87 FE 01:0C
7808:CA F5 76 3A 80 87 FE 01:22
7810:CA 3D 82 3A 78 87 FE 01:B6
7818:CA F5 76 3A A3 86 FE 00:D3
7820:C2 40 78 3A 96 87 FE 01:3C
7828:CA A3 81 3A BA 87 CD 1E:72
7830:BB C2 A3 78 CD B3 78 3A:FB
7838:BE 87 CD 1E BB C4 F8 7F:5A
7840:CD F9 7F CD 26 84 CD 95:41
7848:7D 3E 12 CD 1E BB C4 FE:67
7850:84 3E 42 CD 1E BB C2 3D:F0
7858:82 C3 FB 77 3A A3 86 FE:33
7860:00 C4 B7 82 3A A3 86 FE:1B
7868:00 C2 75 78 3A BE 87 CD:52
7870:1E BB C4 F8 7F CD F9 7F:B8
7878:CD 26 84 CD 95 7D 3E 12:E2
7880:CD 1E BB C4 FE 84 C9 3A:9F
7888:A3 86 FE 00 C0 3A 94 87:09
7890:FE 01 CA 9C 78 21 20 54:74
7898:22 9C 86 C9 21 3C 53 22:82
78A0:9C 86 C9 3A A3 86 FE 00:48
78A8:C2 40 78 3E 01 32 96 87:45
78B0:C3 FB 77 3A A3 86 FE 00:73
78B8:C0 3A BD 87 CD 1E BB C2:99
78C0:FA 7F 3A BC 87 CD 1E BB:E9
78C8:C2 36 81 C9 21 A0 00 3E:6A
78D0:64 E5 E5 36 C0 D1 13 01:30
78D8:09 00 ED B0 E1 11 46 00:95
78E0:19 3D C8 C3 D1 78 CD E8:0F
78E8:83 CD FA 83 3A 95 87 21:D0
78F0:50 00 CD 7F 86 01 3A 91:DD
78F8:09 11 E3 05 06 0A C5 E5:3D
7900:D5 06 08 C5 E5 D5 D5 7E:C0
7908:21 3C 00 CD 7F 86 01 89:3C
7910:59 09 D1 01 06 00 3E 0A:8B
7918:CD B7 85 E1 11 06 00 19:20
7920:EB E1 23 C1 10 DD E1 11:E0
7928:BC 02 19 EB E1 01 08 00:FB
7930:09 C1 10 CA 21 E3 05 11:2E
7938:9F C9 01 30 00 3E 64 C3:ED
7940:35 86 DD 21 D9 86 06 04:C0
7948:DD 36 00 01 11 80 58 DD:B3
7950:73 07 DD 72 08 DD 36 09:C1
7958:05 DD 36 0A 0B CD 52 87:D5
7960:DD 77 0B DD 36 0D 00 DD:7D
7968:36 0E 00 11 11 00 DD 19:9F
7970:10 D6 DD 21 D9 86 DD 36:DC
7978:0C 50 DD 21 EA 86 DD 36:29
7980:0C 3C DD 21 FB 86 DD 36:5E
7988:0C 32 DD 21 0C 87 DD 36:AD
7990:0C 64 DD 21 1D 87 DD 36:6E
7998:00 01 11 69 56 DD 73 07:23
79A0:DD 72 08 DD 36 09 08 DD:CA
79A8:36 0A 10 DD 36 0B 94 DD:53
79B0:36 0C 8E DD 36 0D 00 3E:E1
79B8:01 32 79 87 C9 DD 21 D9:E7
79C0:86 06 04 C5 DD E5 CD E8:75
79C8:79 DD E1 C1 11 11 00 DD:BE
79D0:19 10 F0 CD 55 7A CD 7F:9E
79D8:7A CD C8 7A DD 21 D9 86:DB
79E0:11 11 00 06 04 C3 CB 82:E7
79E8:DD 7E 00 FE 01 CA FB 79:98
79F0:FE 02 CA 0D 7A FE 03 CA:B8
79F8:25 7A C9 DD 7E 0C C6 05:A9
7A00:DD 77 0C FE 96 D8 CD 3F:F2
7A08:87 DD 77 00 C9 3E 02 21:9F
7A10:37 00 01 12 58 CD 99 7F:B9
7A18:DD 7E 0B 3D DD 77 0B FE:D8
7A20:65 D0 C3 3D 7A 3E 02 21:C8
7A28:37 00 01 12 58 CD 99 7F:D1
7A30:DD 7E 0B 3C DD 77 0B FE:EC
7A38:A0 D8 C3 3D 7A CD 52 87:E5
7A40:DD 77 0B DD 36 0C 32 DD:B6
7A48:36 00 01 11 80 58 DD 73:72
7A50:07 DD 72 08 C9 3A 90 87:72
7A58:FE 01 C8 DD 21 1D 87 DD:8A
7A60:7E 0D 3C DD 77 0D FE 01:35
7A68:CA 77 7A FE 02 CA 7B 7A:F3
7A70:DD 36 0D 00 C3 55 7A DD:65
7A78:35 0C C9 DD 34 0C C9 3A:A9
7A80:83 87 FE 00 C8 3A A0 86:59
7A88:FE 91 C0 AF 32 83 87 DD:C3
7A90:21 2E 87 DD 36 00 02 11:34
7A98:B7 58 DD 73 07 DD 72 08:8B
7AA0:DD 36 09 08 DD 36 0A 07:B1
7AA8:DD 35 0B DD 35 0B DD 21:5C
7AB0:82 87 DD 34 00 DD 7E 00:C1
7AB8:FE 09 C2 C2 7A 3E 01 32:FD
7AC0:90 87 CD A2 85 C3 6E 82:04
7AC8:DD 21 2E 87 DD 7E 00 FE:3C
7AD0:02 C0 DD 7E 0C D6 01 DD:8A
7AD8:77 0C FE 55 D0 DD 36 00:E7
7AE0:FF C9 DD 21 2E 87 DD 36:D1
7AE8:00 01 11 5D 56 DD 73 07:44
7AF0:DD 72 08 DD 36 09 01 DD:EA
7AF8:36 0A 0C DD 36 0B 74 DD:B8
7B00:36 0C 95 DD 21 D9 86 DD:49
7B08:36 00 01 11 35 59 DD 73:C2
7B10:07 DD 72 08 DD 36 09 06:C6
7B18:DD 36 0A 18 DD 36 0B 8C:9C
7B20:DD 36 0C 8F DD 36 10 00:49
7B28:AF 32 79 87 C9 DD 21 D9:07
7B30:86 CD 40 7B DD 21 D9 86:AD
7B38:11 11 00 06 01 C3 CB 82:32
7B40:CD 5F 7B FE 00 C8 DD 7E:5A
7B48:10 3C DD 77 10 FE 07 DA:03
7B50:77 7B FE 0B D8 FE 10 DA:CA
7B58:80 7B DD 36 10 01 C9 DD:75
7B60:7E 0B 3D DD 77 0B FE 64:41
7B68:D0 DD 36 0B A0 DD 36 0C:63
7B70:8F DD 36 10 00 AF C9 DD:97
7B78:7E 0C D6 07 DD 77 0C C9:DE
7B80:DD 7E 0C C6 07 DD 77 0C:02
7B88:C9 DD 21 D9 86 06 03 DD:0C
7B90:36 00 01 11 B8 54 DD 73:BB
7B98:07 DD 72 08 DD 36 09 05:46
7BA0:DD 36 0A 17 DD 36 0C A5:EF
7BA8:DD 36 0E 00 11 11 00 DD:39
```

```
7BBØ:19 1Ø DC DD 21 D9 86 DD:B9
7BB8:36 ØB 72 DD 36 1Ø Ø7 DD:DØ
7BCØ:21 EA 86 DD 36 ØB 7E DD:EØ
7BC8:36 1Ø ØF DD 21 FB 86 DD:53
7BDØ:36 ØB 8F DD 36 1Ø ØA DD:54
7BD8:21 ØC 87 DD 36 ØØ Ø1 11:32
7BEØ:2B 55 DD 73 Ø7 DD 72 Ø8:42
7BE8:DD 36 Ø9 Ø3 DD 36 ØA 11:36
7BFØ:DD 36 ØB 8C DD 36 ØC 78:AE
7BF8:DD 36 ØD ØØ DD 36 1Ø ØØ:E8
7CØØ:AF 32 79 87 C9 DD 21 D9:EØ
7CØ8:86 Ø6 Ø3 C5 DD E5 CD 2A:CD
7C1Ø:7C DD E1 C1 11 11 ØØ DD:ØC
7C18:19 1Ø FØ CD A1 7C DD 21:F1
7C2Ø:D9 86 11 11 ØØ Ø6 Ø4 C3:5Ø
7C28:CB 82 DD 7E ØØ FE ØØ C8:36
7C3Ø:FE Ø1 CA 45 7C FE Ø2 CA:DC
7C38:58 7C FE Ø3 CA 73 7C FE:Ø2
7C4Ø:Ø4 CA 86 7C C9 DD 7E 1Ø:E3
7C48:3D DD 77 1Ø FE ØØ CØ DD:7E
7C5Ø:36 ØØ Ø2 DD 36 1Ø ØA C9:78
7C58:DD 7E ØC D6 Ø5 DD 77 ØC:11
7C6Ø:DD 7E 1Ø 3D DD 77 1Ø FE:54
7C68:ØØ CØ DD 36 ØØ Ø3 DD 36:AØ
7C7Ø:1Ø Ø5 C9 DD 7E 1Ø 3D DD:3E
7C78:77 1Ø FE ØØ CØ DD 36 ØØ:ED
7C8Ø:Ø4 DD 36 1Ø ØA C9 DD 7E:7F
7C88:ØC C6 Ø5 DD 77 ØC DD 7E:B5
7C9Ø:1Ø 3D DD 77 1Ø FE ØØ CØ:4D
7C98:DD 36 ØØ Ø1 DD 36 1Ø Ø4:86
7CAØ:C9 DD 21 ØC 87 3E Ø6 21:7B
7CA8:33 ØØ Ø1 2B 55 CD 99 7F:A4
7CBØ:CD EF 7C DD 7E ØØ FE Ø1:2F
7CB8:CA C1 7C FE Ø2 CA D8 7C:7A
7CCØ:C9 DD 7E ØB 3D DD 77 ØB:5D
7CC8:FE 66 DØ DD 36 ØØ Ø2 DD:F6
7CDØ:36 ØC 78 DD 36 1Ø ØØ C9:2C
7CD8:DD 7E ØB 3C DD 77 ØB FE:96
7CEØ:AØ D8 DD 36 ØØ Ø1 DD 36:DC
7CE8:ØC 78 DD 36 1Ø ØØ C9 DD:86
7CFØ:7E 1Ø 3C DD 77 1Ø FE Ø5:FF
7CF8:DA Ø8 7D FE Ø7 D8 FE ØB:4A
7DØØ:DA 11 7D DD 36 1Ø ØØ C9:1A
7DØ8:DD 7E ØC D6 Ø2 DD 77 ØC:B3
7D1Ø:C9 DD 7E ØC C6 Ø2 DD 77:67
7D18:ØC C9 DD 21 95 86 D5 1A:FE
7D2Ø:11 11 ØØ FE Ø1 28 Ø5 DD:C8
7D28:19 3D 18 F7 D1 C9 3E Ø2:E9
7D3Ø:CD 9Ø BB 26 Ø1 2E Ø1 CD:EB
7D38:75 BB CD ØØ B9 3A AØ 86:9Ø
7D4Ø:6F 26 ØØ CD 44 EF C3 Ø3:Ø7
7D48:B9 3A 85 87 FE ØØ C8 3A:DB
7D5Ø:86 87 3D FE ØØ CA 71 7D:CB
7D58:32 86 87 3A 88 87 3C 32:96
7D6Ø:88 87 FE Ø1 C8 FE Ø2 CA:AB
7D68:89 7D AF 32 88 87 C3 5B:3C
7D7Ø:7D AF 32 85 87 C3 87 78:18
7D78:3A 8F 87 FE Ø1 C8 3A 94:C5
7D8Ø:87 FE Ø1 CA 1B 81 C3 ØØ:8D
7D88:81 21 98 3A 22 9C 86 3E:64
7D9Ø:Ø1 32 8F 87 C9 CD 49 7D:DD
7D98:3A 79 87 47 FE ØØ C4 59:ØC
7DAØ:85 3A 8F 87 FE ØØ CC 87:A1
7DA8:78 AF 32 8F 87 32 A5 86:4F
7DBØ:11 8B 86 Ø6 ØA CD 1A 7D:9C
7DB8:C5 D5 DD 7E ØØ FE ØØ 28:67
7DCØ:44 DD 7E ØC FE D2 D2 Ø5:AD
7DC8:7E DD 7E ØB FE AØ D2 Ø5:BF
7DDØ:7E CD 33 7F FE Ø1 CA Ø5:A4
7DD8:7E 11 38 4A CD B4 7F DD:5F
7DEØ:75 Ø1 DD 74 Ø2 11 C8 4B:7B
7DE8:CD B4 7F DD 75 Ø3 DD 74:91
7DFØ:Ø4 DD 56 Ø6 DD 5E Ø5 DD:D5
7DF8:4E Ø9 Ø6 ØØ DD 7E ØA CD:DA
7EØØ:1E 7F CD CF 85 D1 C1 13:9B
7EØ8:1Ø AB 11 8B 86 Ø6 ØA CD:BB
7E1Ø:1A 7D C5 D5 DD 7E ØØ FE:7A
7E18:ØØ 28 3C FE FF CA 57 7E:9A
7E2Ø:DD 7E ØC FE D2 D2 57 7E:EA
7E28:DD 7E ØB FE AØ D2 57 7E:F5
7E3Ø:CD 33 7F FE Ø1 CA 57 7E:68
7E38:DD 66 Ø4 DD 6E Ø3 DD 56:D2
7E4Ø:Ø8 DD 5E Ø7 DD 46 Ø9 CD:52
7E48:59 7F CD 78 7F DD 7E ØA:CF
7E5Ø:CD 1E 7F 4F CD FA 85 D1:98
7E58:C1 13 1Ø B3 21 E3 Ø5 11:5B
7E6Ø:9F C9 Ø1 3Ø ØØ 3E 64 CD:6A
7E68:35 86 C3 BC 7E 11 8B 86:39
7E7Ø:Ø6 ØA CD 1A 7D C5 D5 DD:A1
7E78:7E ØØ FE ØØ 28 39 DD 7E:87
7E8Ø:ØC FE D2 D2 B7 7E DD 7E:46
7E88:ØB FE AØ D2 B7 7E CD 33:EF
7E9Ø:7F FE Ø1 CA B7 7E DD 66:76
7E98:Ø4 DD 6E Ø3 2B DD 56 Ø2:99
7EAØ:DD 5E Ø1 DD 7E Ø9 C6 Ø2:54
7EA8:4F CD 43 7F Ø6 ØØ DD 7E:ED
7EBØ:ØA CD 1E 7F CD 35 86 D1:99
7EB8:C1 13 1Ø B6 11 8B 86 Ø6:96
7ECØ:ØA CD 1A 7D C5 D5 DD 7E:F6
7EC8:ØØ FE ØØ 28 3E DD 7E ØC:18
7EDØ:FE D2 D2 ØB 7F DD 7E ØB:Ø5
7ED8:FE AØ D2 ØB 7F CD 33 7F:DC
7EEØ:FE Ø1 CA ØB 7F DD 7E ØØ:Ø3
7EE8:FE FF C2 F1 7E DD 36 ØØ:8A
7EFØ:ØØ DD 66 Ø6 DD 6E Ø5 DD:62
7EF8:56 Ø4 DD 5E Ø3 DD 4E Ø9:8A
7FØØ:Ø6 ØØ DD 7E ØA CD 1E 7F:DE
7FØ8:CD B7 85 D1 C1 13 1Ø B1:C4
7F1Ø:C9 DD 7E ØB FE 65 D2 1B:A2
7F18:7F AF C9 D6 64 C9 F5 DD:6C
7F2Ø:7E ØC DD 86 ØA FE D2 D2:58
7F28:2C 7F F1 C9 F1 3E D2 DD:97
7F3Ø:96 ØC C9 3E 64 DD 96 Ø9:34
7F38:DD BE ØB DA 41 7F 3E Ø1:92
7F4Ø:C9 AF C9 DD 7E ØB DD 86:A8
7F48:Ø9 FE AØ D2 4F 7F C9 Ø6:28
7F5Ø:AØ 9Ø 47 DD 7E Ø9 9Ø 4F:EC
7F58:C9 DD 7E ØB DD 86 Ø9 FE:A4
7F6Ø:AØ D2 6B 7F DD 7E ØA DD:D3
7F68:46 Ø9 C9 Ø6 AØ 9Ø 47 DD:ØB
7F7Ø:7E Ø9 9Ø 47 DD 7E ØA C9:1E
7F78:DD 7E ØB FE 65 DA 84 7F:92
7F8Ø:DD 46 Ø9 C9 3E 64 DD 96:FØ
7F88:ØB F5 E5 6F 26 ØØ 19 EB:2C
7F9Ø:E1 47 DD 7E Ø9 9Ø 47 F1:13
7F98:C9 32 A1 7F DD 7E ØD 3C:A3
7FAØ:FE ØØ C2 A6 7F AF DD 77:53
7FA8:ØD CD 7F 86 Ø9 DD 75 Ø7:29
7FBØ:DD 74 Ø8 C9 DD 7E ØC D6:79
7FB8:32 6F 26 ØØ 29 19 5E 23:C6
7FCØ:56 CD 11 7F 6F 26 ØØ 19:35
7FC8:C9 Ø6 C7 21 A8 E8 11 38:E4
7FDØ:4A C5 7D 12 13 7C 12 13:3F
7FD8:D5 CD 26 BC D1 C1 1Ø F1:BB
7FEØ:C9 Ø6 C7 21 64 ØØ 11 C8:B8
7FE8:4B C5 7D 12 13 7C 12 13:58
7FFØ:Ø1 46 ØØ Ø9 C1 1Ø F2 C9:2B
7FF8:C9 C9 3A 94 87 FE Ø1 CA:BE
8ØØØ:ØC 8Ø DD 21 95 86 DD 35:67
8ØØ8:ØB C3 3E 8Ø 3A 95 87 FE:14
8Ø1Ø:ØØ C2 3E 8Ø DD 21 95 86:28
8Ø18:DD 7E ØC FE 8E C2 3E 8Ø:91
8Ø2Ø:DD 7E ØB FE 7Ø C2 3E 8Ø:ØØ
8Ø28:3A 83 87 FE Ø1 CA 3E 8Ø:E8
8Ø3Ø:3E Ø1 32 83 87 CD 7A 8Ø:59
8Ø38:CD 94 85 CD 6E 82 AF 32:FB
8Ø4Ø:A2 86 3E Ø1 32 94 87 32:DF
8Ø48:8F 87 3A AØ 86 3C FE 9A:5B
8Ø5Ø:D4 D4 8Ø 32 AØ 86 CD 7A:43
8Ø58:8Ø C3 1B 81 3A 83 87 FE:Ø8
8Ø6Ø:ØØ C8 FD 21 95 86 DD 21:ØB
8Ø68:2E 87 FD 7E ØB 3D DD 77:7B
8Ø7Ø:ØB FD 7E ØC C6 Ø3 DD 77:52
8Ø78:ØC C9 3A 83 87 FE ØØ C8:27
8Ø8Ø:FD 21 95 86 DD 21 2E 87:A7
8Ø88:FD 7E ØB C6 Ø4 DD 77 ØB:15
8Ø9Ø:FD 7E ØC C6 Ø3 DD 77 ØC:23
8Ø98:C9 3A 95 87 FE ØØ CA B7:64
8ØAØ:8Ø 3D 32 95 87 CD E6 78:7F
8ØA8:DD 21 2E 87 DD 7E ØØ FE:22
8ØBØ:Ø2 CC BA 8Ø 3E 99 C9 3E:33
8ØB8:69 C9 DD 36 ØØ FF C9 3A:EB
8ØCØ:95 87 FE Ø1 CØ 3A 83 87:CA
8ØC8:FE Ø1 C8 DD 21 2E 87 DD:66
8ØDØ:36 ØØ FF C9 3A ØA 85 FE:98
8ØD8:Ø1 CA F1 8Ø 3A 95 87 3C:F1
8ØEØ:FE Ø3 CA F1 8Ø 32 95 87:7D
8ØE8:CD E6 78 CD BF 8Ø 3E 69:52
8ØFØ:C9 3E 9A C9 E5 78 CD AD:F3
8ØF8:BC E1 FE 84 C8 C3 AA BC:ØØ
81ØØ:3A A2 86 3C FE Ø3 C2 ØA:27
81Ø8:81 AF 32 A2 86 21 4C ØØ:FE
811Ø:CD 7F 86 Ø1 D4 53 Ø9 22:57
8118:9C 86 C9 3A A4 86 3C FE:7Ø
812Ø:Ø3 C2 25 81 AF 32 A4 86:DE
8128:21 4C ØØ CD 7F 86 Ø1 FØ:BC
813Ø:52 Ø9 22 9C 86 C9 3A 94:75
8138:87 FE ØØ CA 48 81 DD 21:E5
814Ø:95 86 DD 34 ØB C3 7A 81:FØ
8148:3A 95 87 FE ØØ C2 7A 81:A4
815Ø:DD 21 95 86 DD 7E ØC FE:5Ø
8158:8E C2 7A 81 DD 7E ØB FE:DF
816Ø:75 C2 7A 81 3A 83 87 FE:21
8168:Ø1 CA 7A 81 3E Ø1 32 83:A2
817Ø:87 CD 5C 8Ø CD 94 85 CD:AA
8178:6E 82 AF 32 A4 86 32 94:96
818Ø:87 3E Ø1 32 8F 87 3A AØ:5A
8188:86 3D FE 69 DC 99 8Ø 32:99
819Ø:AØ 86 CD 5C 8Ø C3 ØØ 81:AE
8198:3A 94 87 FE Ø1 CA 7A 8Ø:1F
81AØ:C3 5C 8Ø 3A AØ 86 32 87:DE
81A8:87 Ø6 Ø7 C5 CD B3 78 CD:C8
81BØ:98 81 3E 42 CD 1E BB C2:6F
81B8:3C 82 3A 8Ø 87 FE Ø1 CA:15
81CØ:3C 82 3A 96 87 FE ØØ CA:6E
81C8:35 82 3A A1 86 D6 Ø3 32:FB
81DØ:A1 86 CD 5C 78 C1 1Ø D3:BB
81D8:Ø6 Ø2 C5 CD B3 78 CD 98:9Ø
81EØ:81 3E 42 CD 1E BB C2 3C:7E
81E8:82 3A 8Ø 87 FE Ø1 CA 3C:5D
81FØ:82 3A 96 87 FE ØØ CA 35:69
81F8:82 CD 5C 78 C1 1Ø DB Ø6:DB
82ØØ:Ø7 C5 CD B3 78 CD 98 81:9C
82Ø8:3E 42 CD 1E BB C2 3C 82:12
821Ø:3A 8Ø 87 FE Ø1 CA 3C 82:CE
8218:3A 96 87 FE ØØ CA 35 82:CC
822Ø:3A A1 86 C6 Ø3 32 A1 86:9A
8228:CD 5C 78 C1 1Ø D3 AF 32:36
823Ø:96 87 C3 FB 77 E1 CD 5C:9F
8238:78 C3 FB 77 E1 CD 76 85:1A
824Ø:21 64 ØØ 11 9F C9 Ø1 3Ø:47
8248:ØØ 3E 64 CD 39 84 21 2C:22
825Ø:89 11 3E EB CD 52 84 Ø6:9C
8258:FA CD 59 85 Ø6 FA CD 59:EA
826Ø:85 Ø6 FA CD 59 85 Ø6 FA:6A
8268:CD 59 85 C3 6Ø 75 DD 21:B5
827Ø:B8 87 Ø6 Ø5 DD 7E ØØ FE:13
8278:ØA 38 Ø9 ØE ØA 91 DD 77:22
828Ø:ØØ DD 34 FF DD 2B 1Ø EC:77
8288:DD 21 B5 87 11 69 D6 Ø6:39
829Ø:Ø4 C5 D5 DD E5 D5 DD 7E:Ø5
8298:ØØ 21 1Ø ØØ CD 7F 86 Ø1:39
82AØ:5Ø 4E Ø9 D1 Ø1 Ø2 ØØ 3E:6E
82A8:Ø8 CD 39 84 DD E1 D1 C1:DD
82BØ:DD 23 13 13 1Ø DB C9 3E:BB
82B8:Ø1 32 85 87 3E 14 32 86:86
82CØ:87 AF 32 A3 86 32 88 87:Ø3
82C8:C3 CE 76 ED 53 8A 87 F5:F3
82DØ:C5 DD E5 C5 DD E5 16 ØØ:DD
82D8:DD 7E ØØ FE ØØ CA 34 83:6B
82EØ:DD 7E ØE FE ØØ C2 34 83:6D
82E8:3A A3 86 FE ØØ C2 34 83:84
82FØ:3A 85 87 FE Ø1 CA 34 83:88
82F8:3A AØ 86 C6 Ø3 DD BE ØB:65
83ØØ:D4 E6 83 3A AØ 86 DD 96:93
83Ø8:Ø9 3C DD BE ØB DC E6 83:5C
831Ø:3A A1 86 C6 11 DD BE ØC:CE
8318:D4 E6 83 3A A1 86 DD 96:BØ
832Ø:ØA C6 Ø4 DD BE ØC DC E6:EB
8328:83 7A FE Ø4 C2 34 83 3E:B3
```

```
8330:01 32 A3 86 DD E1 C1 ED:5F
8338:5B 8A 87 DD 19 10 94 DD:04
8340:E1 C1 F1 C9 C5 DD E5 DD:4F
8348:7E 00 FE 00 CA AD 83 DD:C0
8350:7E 0E FE 00 C2 AD 83 FD:BC
8358:21 A6 86 06 03 16 00 FD:6D
8360:7E 00 FE 00 CA A6 83 FD:AE
8368:7E 0E FE 00 C2 A6 83 DD:AA
8370:7E 0B DD 86 09 FD BE 0B:DB
8378:D4 E6 83 DD 7E 0B D6 02:3A
8380:FD BE 0B DC E6 83 DD 7E:98
8388:0C DD 86 0A FD BE 0C D4:E4
8390:E6 83 DD 7E 0C D6 03 FD:CB
8398:BE 0C DC E6 83 7A FE 04:9A
83A0:C2 A6 83 CD B9 83 11 11:9C
83A8:00 FD 19 10 B0 DD E1 C1:7D
83B0:ED 5B 8A 87 DD 19 10 8C:47
83B8:C9 FD 36 0E 01 DD 7E 0F:F5
83C0:3D DD 77 0F FE 00 C2 D5:C7
83C8:83 DD 36 0E 01 FD 36 0E:3F
83D0:00 FD 36 00 FF CD 84 85:7C
83D8:DD E5 FD E5 C5 CD 6E 82:46
83E0:C1 FD E1 DD E1 C9 14 C9:24
83E8:DD 21 A6 86 06 08 DD 36:9D
83F0:00 00 11 11 00 DD 19 10:47
83F8:F5 C9 3A 95 87 FE 00 CA:EB
8400:0D 84 FE 01 CA 16 84 FE:99
8408:02 CA 1E 84 C9 3E 01 32:84
8410:8C 87 CD E2 7A C9 3E 02:F7
8418:32 8C 87 C3 89 7B 3E 03:E0
8420:32 8C 87 C3 42 79 3A 8C:A5
8428:87 FE 01 CA 2D 7B FE 02:1F
8430:CA 05 7C FE 03 CA BD 79:B2
8438:C9 C5 E5 D5 ED B0 E1 11:82
8440:00 08 19 30 04 11 50 C0:89
8448:19 EB E1 C1 09 3D C8 C3:8D
8450:39 84 7E FE 20 C2 5A 84:49
8458:3E 2D D6 2D E5 D5 D5 21:FC
8460:10 00 CD 7F 86 01 20 4E:4B
8468:09 01 02 00 3E 08 D1 CD:82
8470:AC 84 D1 E1 13 13 23 7E:55
8478:FE 2A C2 52 84 23 C9 7E:B1
8480:FE 20 C2 87 84 3E 2D D6:97
8488:2D E5 D5 D5 21 10 00 CD:43
8490:7F 86 01 20 4E 09 01 02:F5
8498:00 3E 08 D1 CD 39 84 D1:6F
84A0:E1 13 13 23 7E FE 2A C2:90
84A8:7F 84 23 C9 C5 E5 C5 E5:02
84B0:D5 ED B0 D1 21 00 08 19:DC
84B8:30 04 11 50 C0 19 EB E1:B2
84C0:C1 D5 ED B0 D1 21 00 08:51
84C8:19 30 04 11 50 C0 19 EB:2C
84D0:E1 C1 09 3D C8 C3 AC 84:14
84D8:AF F5 06 00 0E 00 CD 32:78
84E0:BC F1 3C FE 10 C8 C3 D9:CB
84E8:84 21 B6 89 AF E5 F5 46:24
84F0:4E CD 32 BC F1 E1 3C FE:71
84F8:10 C8 23 C3 ED 84 CD 06:15
8500:BB CD 9C 77 CD 06 BB C3:E4
8508:06 BB 00 3E 01 32 94 87:76
8510:AF 32 A3 86 32 96 87 32:68
8518:A4 86 32 A2 86 32 8F 87:56
8520:DD 21 95 86 DD 36 00 01:38
8528:DD 36 0B 78 DD 36 0C 8E:50
8530:DD 36 09 04 DD 36 0A 13:9C
8538:21 F0 52 DD 75 07 DD 74:46
8540:08 C9 3E 0A 32 79 87 C9:0A
8548:AF 32 B5 87 32 B6 87 32:9A
8550:B7 87 32 B8 87 32 B9 87:26
8558:C9 78 FE 00 C8 21 F4 01:F2
8560:2B 7C B5 20 FB 10 F6 C9:F0
8568:11 64 00 21 A8 E8 01 46:39
8570:00 3E A0 C3 E4 85 21 64:F6
8578:00 36 00 11 65 00 01 50:2D
8580:46 ED B0 C9 C3 94 85 3A:13
8588:B7 87 3C 32 B7 87 11 01:8A
8590:00 C3 AF 85 3A B7 87 C6:09
8598:02 32 B7 87 11 02 00 C3:3D
85A0:AF 85 3A B5 87 3C 32 B5:71
85A8:87 11 64 00 C3 AF 85 2A:DE
85B0:7E 87 19 22 7E 87 C9 ED:1B
85B8:43 BC 85 01 00 00 F5 D5:E6
85C0:ED B0 D1 E5 21 46 00 19:AA
85C8:EB E1 F1 3D C8 18 EC ED:15
85D0:43 D4 85 01 00 00 F5 E5:AE
85D8:ED B0 E1 01 46 00 09 F1:76
85E0:3D C8 18 EF C5 F5 E5 ED:78
85E8:B0 E1 01 00 08 09 30 04:B0
85F0:01 50 C0 09 F1 C1 3D C8:A0
85F8:18 EA C5 E5 D5 1A FE 00:03
8600:28 1D 7E FE 00 28 13 E6:FF
8608:55 28 0A 7E E6 AA 20 0F:1B
8610:1A E6 AA 18 06 1A E6 55:86
8618:18 01 1A 4F 7E B1 77 13:B7
8620:23 10 DA E1 16 00 DD 5E:64
8628:09 19 EB E1 01 46 00 09:1F
8630:C1 0D C8 18 C5 ED 43 3A:55
8638:86 01 00 00 E5 D5 ED B0:B8
8640:D1 21 00 08 19 30 04 11:3A
8648:50 C0 19 EB E1 01 46 00:EA
8650:09 3D C8 18 E4 21 4A C9:99
8658:CD 11 7F 4F 06 00 09 DD:CB
8660:7E 0C D6 32 FE 00 CA 78:02
8668:86 11 00 08 19 D2 74 86:7B
8670:11 50 C0 19 3D C2 69 86:17
8678:DD 74 02 DD 75 01 C9 EB:63
8680:21 00 00 FE 00 C8 19 3D:66
8688:C3 83 86 01 02 03 04 05:CD
8690:06 07 08 09 0A 01 00 00:9E
8698:00 00 2A 92 00 00 0A 1A:FA
86A0:00 00 00 00 00 00 00 00:26
86A8:00 00 00 EA 92 00 00 00:B0
86B0:00 00 00 00 00 00 00 00:36
86B8:00 00 00 00 11 93 00 00:05
86C0:00 00 00 00 00 00 00 00:46
86C8:00 00 00 00 00 38 93 00:A3
86D0:00 00 00 00 00 00 00 00:56
86D8:00 00 00 00 00 00 5F 93:8F
86E0:00 00 00 00 00 00 00 00:66
86E8:00 00 00 00 00 00 00 1F:66
86F0:94 00 00 00 00 00 00 00:0A
86F8:00 00 00 00 00 00 00 00:7E
8700:DF 94 00 00 00 00 00 00:8E
8708:00 00 00 00 00 00 00 00:8F
8710:00 9F 95 00 00 00 00 00:94
8718:00 00 00 00 00 00 00 00:9F
8720:00 00 5F 96 00 00 00 00:1C
8728:00 00 00 00 00 00 00 00:AF
8730:00 00 00 1F 97 00 00 00:26
8738:00 00 00 00 00 00 00 2A:0F
8740:AA 87 23 22 AA 87 7E FE:4E
8748:00 C0 21 AC 87 22 AA 87:AF
8750:7E C9 2A 98 87 23 22 98:E8
8758:87 7E FE 00 C0 21 9A 87:50
8760:7E 22 98 87 C9 2A A2 87:1C
8768:23 22 A2 87 7E FE 00 C0:C2
8770:21 A4 87 7E 22 A2 87 C9:5C
8778:00 00 00 00 00 00 00 00:FF
8780:00 00 00 00 00 00 00 00:07
8788:00 00 00 00 00 00 00 00:0F
8790:00 00 00 00 00 00 00 00:17
8798:00 00 73 91 82 78 7D 96:31
87A0:6D 00 00 00 5F 58 51 00:B6
87A8:00 00 00 00 02 03 03 02:70
87B0:03 02 02 03 00 00 00 00:50
87B8:00 00 00 00 00 00 00 E5:67
87C0:D2 43 4F 4E 47 52 41 54:7A
87C8:55 4C 41 54 49 4F 4E 53:50
87D0:20 2F 2F 2A 83 F3 59 4F:32
87D8:55 52 20 4D 49 53 53 49:D8
87E0:4F 4E 20 49 53 20 4F 56:0E
87E8:45 52 2A 46 EA 50 52 4F:16
87F0:47 52 41 4D 20 41 4E 44:C1
87F8:20 4D 55 53 49 43 2A 97:61
8800:EA 42 59 20 4C 55 43 20:D0
8808:47 55 49 4C 4C 41 55 4D:49
8810:45 2A 3B EB 47 52 41 50:24
8818:48 49 43 53 20 42 59 2A:7A
8820:88 EB 48 45 52 56 45 20:73
8828:47 55 49 4C 4C 41 55 4D:69
8830:45 2A 2B EC 44 45 53 49:01
8838:47 4E 20 31 39 39 30 2A:DA
8840:71 EC 43 4F 50 59 52 49:42
8848:47 48 54 20 42 4C 41 43:14
8850:4B 20 53 59 53 54 45 4D:A2
8858:2A A8 E8 00 51 04 00 0B:17
8860:00 E2 E8 2C 51 04 00 0B:19
8868:00 98 DE 58 51 04 00 0B:1F
8870:00 D2 DE 84 51 04 00 0B:4B
8878:00 0A F6 2E 52 03 00 0A:AA
8880:00 0F F6 2E 52 03 00 0A:BC
8888:00 B0 F5 4C 52 06 00 16:ED
8890:00 D9 F5 4C 52 06 00 16:47
8898:00 0E E1 B0 51 12 00 07:D8
88A0:00 9F C9 10 27 30 00 64:04
88A8:00 00 C0 10 27 50 00 15:FB
88B0:00 E0 F6 10 27 50 00 12:4D
88B8:00 00 C0 10 27 08 00 C8:F3
88C0:00 46 C0 10 27 0A 00 C8:93
88C8:00 A9 E8 00 51 04 00 0B:5F
88D0:00 E2 E8 2C 51 04 00 0B:89
88D8:00 B9 C5 43 52 45 44 49:89
88E0:54 53 20 33 2A D0 C5 50:23
88E8:4F 57 45 52 20 3E 3E 3E:3A
88F0:3E 2A 59 DE 42 4C 41 43:7E
88F8:4B 20 53 59 53 54 45 4D:4A
8900:20 31 39 39 30 2A 30 30:56
8908:30 30 30 30 2A 32 2A 30:15
8910:30 2A 43 4F 50 59 52 49:4E
8918:47 48 54 20 41 43 4B 45:00
8920:52 4C 49 47 48 54 20 31:52
8928;39 38 39 2A 47 41 4D 45:D9
8930:20 4F 56 45 52 2A 3E 20:D5
8938:4D 49 53 53 49 4F 4E 20:4E
8940:4F 56 45 52 20 3C 2A 4C:69
8948:45 56 45 4C 20 31 2A 3F:A5
8950:3F 3F 3F 3F 3F 3F 2A 50:AA
8958:4C 45 41 53 45 20 41 43:BE
8960:43 45 53 53 20 43 4F 44:76
8968:45 20 3D 3D 3D 3D 3D 2A:BB
8970:3F 2A 3E 3E 3E 3E 2A 02:1E
8978:06 00 0A 00 0A 00 14 00:E3
8980:02 01 03 14 00 00 0F 0F:47
8988:00 01 00 00 00 00 01 0F:92
8990:02 00 01 00 00 00 00 06:4E
8998:0F 02 00 02 00 00 00 00:3C
89A0:0C 0F 02 00 02 00 00 14:03
89A8:00 00 0F 05 00 02 03 03:AB
89B0:EE 02 19 0F 32 00 00 00:AC
89B8:06 02 01 18 17 16 0A 09:33
89C0:0F 10 12 1A 0E 03 DD 7E:69
89C8:0E 3C DD 77 0E FE 03 DA:69
89D0:F4 89 FE 04 DA FD 89 FE:48
89D8:05 DA 0F 8A FE 06 DA 19:47
89E0:8A FE 07 DA 0F 8A FE 08:25
89E8:DA FD 89 DD 36 0E 00 DD:9E
89F0:36 00 FF C9 21 10 27 DD:CE
89F8:75 07 DD 74 08 21 91 63:68
8A00:DD 75 07 DD 74 08 DD 36:09
8A08:09 03 DD 36 0A 0D C9 21:17
8A10:B8 63 DD 75 07 DD 74 08:40
8A18:C9 21 DF 63 DD 75 07 DD:FE
8A20:74 08 C9 12 00 01 01 01:E6
8A28:03 02 02 64 02 64 02 01:C7
8A30:01 01 03 04 00 1E 00 43:A2
8A38:00 F8 01 09 02 06 01 7E:FE
8A40:02 01 00 00 0E 0D 00 00:62
8A48:06 00 00 00 06 01 00 00:FC
8A50:0E 0D 00 00 06 01 00 00:26
8A58:0E 0D 00 00 06 00 00 00:28
8A60:06 00 00 00 06 FF 01 CC:6F
8A68:02 0E 05 00 04 B3 00 4A:B5
8A70:0E 06 01 00 00 0E 16 00:05
8A78:00 06 01 00 00 0E 16 00:FF
8A80:00 06 01 00 00 0E 02 00:7B
8A88:00 06 01 00 00 0E 0D 00:D0
8A90:00 06 01 7E 02 0E 05 00:A2
8A98:02 50 00 0E 15 06 01 38:50
8AA0:02 0E 05 02 47 00 0E 15:CC
8AA8:04 8E 00 4A 12 06 00 00:F8
```

```
8AB0:00 06 00 00 00 06 01 00:71
8AB8:00 0E 02 00 00 06 01 00:8F
8AC0:00 0E 0D 00 00 06 01 18:78
8AC8:02 0E 05 00 02 86 00 0E:1D
8AD0:15 06 00 00 00 06 00 00:9F
8AD8:00 06 01 18 02 0E 05 00:52
8AE0:02 86 00 0E 15 06 01 00:44
8AE8:00 0E 02 00 00 06 01 00:BF
8AF0:00 0E 0D 00 00 06 01 18:A8
8AF8:02 0E 05 00 02 43 00 0E:BB
8B00:15 06 01 38 02 0E 05 00:10
8B08:02 47 00 0E 15 06 01 00:EF
8B10:00 0E 16 00 00 06 01 7E:14
8B18:02 0E 05 02 00 00 0E 16:EA
8B20:04 50 00 0E 15 06 01 CC:7B
8B28:02 0E 05 02 00 00 0E 02:5A
8B30:04 59 00 0E 15 06 01 00:3D
8B38:00 0E 0D 00 00 06 00 00:2A
8B40:00 06 01 7E 02 0E 05 00:53
8B48:04 9F 00 4A 0E 06 00 00:A7
8B50:00 06 00 00 00 06 01 00:12
8B58:00 0E 02 00 00 06 01 00:30
8B60:00 0E 0D 00 00 06 01 38:19
8B68:02 0E 05 00 02 47 00 0E:44
8B70:15 06 01 7E 02 0E 05 02:A8
8B78:50 00 0E 15 04 9F 00 4A:EF
8B80:0F 06 01 00 00 0E 16 00:17
8B88:00 06 01 00 00 0E 16 00:10
8B90:00 06 01 00 00 0E 02 00:8C
8B98:00 06 01 00 00 0E 0D 00:E1
8BA0:00 06 01 A4 02 0E 05 00:4B
8BA8:02 54 00 0E 15 05 01 7E:93
8BB0:02 0E 05 02 50 00 0E 15:0A
8BB8:00 05 01 EF 00 0E 05 00:83
8BC0:02 77 00 4A 12 06 00 00:E1
8BC8:00 06 00 00 00 06 01 00:8A
8BD0:00 0E 02 00 00 06 01 00:A8
8BD8:00 0E 0D 00 00 06 01 D5:79
8BE0:00 0E 05 00 04 6A 00 0E:96
8BE8:15 06 01 EF 00 0E 05 00:CA
8BF0:04 77 00 4A 12 06 01 00:1A
8BF8:00 0E 16 00 00 06 01 00:0C
8C00:00 0E 16 00 00 06 01 00:15
8C08:00 0E 02 00 00 06 01 00:E1
8C10:00 0E 0D 00 00 06 00 00:03
8C18:00 06 FF 01 00 00 0E 02:23
8C20:00 00 06 00 00 00 06 00:E8
8C28:00 00 06 FF 01 CC 02 0E:0D
8C30:05 00 04 B3 00 4A 0E 06:E7
8C38:01 00 00 0E 16 00 00 06:9B
8C40:01 00 00 0E 16 00 00 06:A3
8C48:01 00 00 0E 02 00 00 06:47
8C50:01 00 00 0E 0D 00 00 06:86
8C58:01 CC 02 0E 05 00 04 B3:88
8C60:00 5E 0F 06 00 00 00 06:1D
8C68:00 00 00 06 00 00 00 06:3C
8C70:01 00 00 0E 0D 00 00 06:A6
8C78:01 00 00 0E 02 00 00 06:77
8C80:00 00 00 06 00 00 00 06:54
8C88:01 7E 02 0E 05 00 04 9F:7C
8C90:00 4A 0E 06 01 00 00 0E:67
8C98:16 00 00 06 01 00 00 0E:C7
8CA0:16 00 00 06 01 00 00 0E:CF
8CA8:02 00 00 06 01 00 00 0E:C3
8CB0:0D 00 00 06 01 7E 02 0E:D8
8CB8:05 00 04 9F 00 5E 0F 06:9E
8CC0:00 00 00 06 00 00 00 06:94
8CC8:00 00 00 06 01 00 00 0E:E1
8CD0:0D 00 00 06 01 00 00 0E:F6
8CD8:02 00 00 06 00 00 00 06:AE
8CE0:00 00 00 06 01 38 02 0E:57
8CE8:05 00 04 8E 00 4A 0E 06:0B
8CF0:01 00 00 0E 16 00 00 06:53
8CF8:01 00 00 0E 16 00 00 06:5B
8D00:01 00 00 0E 02 00 00 06:00
8D08:01 00 00 0E 0D 00 00 06:3F
8D10:01 38 02 0E 05 00 04 8E:F1
8D18:00 5E 0F 06 00 00 00 06:D6
8D20:00 00 00 06 00 00 00 06:F5
8D28:01 00 00 0E 0D 00 00 06:5F
8D30:01 00 00 0E 02 00 00 06:30
8D38:00 00 00 06 00 00 00 06:0D
8D40:01 18 02 0E 05 00 04 86:A1
8D48:00 4A 0E 06 01 00 00 0E:20
8D50:16 00 00 06 01 00 00 0E:80
8D58:16 00 00 06 01 00 00 0E:88
8D60:02 00 00 06 01 00 00 0E:7C
8D68:0D 00 00 06 01 18 02 0E:2D
8D70:05 00 02 86 00 5E 0E 06:E6
8D78:00 00 00 06 00 00 00 06:4D
8D80:00 00 00 06 01 00 00 0E:9A
8D88:0D 00 00 06 01 00 00 0E:AF
8D90:02 00 00 06 00 00 00 06:67
8D98:01 53 03 0E 05 00 04 6A:92
8DA0:00 0E 15 06 01 18 02 0E:B3
8DA8:05 00 04 43 00 0E 15 06:69
8DB0:00 00 00 06 01 18 02 0E:68
8DB8:05 00 04 43 00 0E 15 06:79
8DC0:01 38 02 0E 05 00 04 47:69
8DC8:00 0E 15 06 00 00 00 06:F8
8DD0:01 38 02 0E 05 00 04 47:79
8DD8:00 0E 15 06 01 7E 02 0E:4F
8DE0:05 00 04 50 00 0E 15 06:D5
8DE8:00 00 00 06 01 38 02 0E:60
8DF0:05 00 04 47 00 0E 15 06:C1
8DF8:01 7E 02 0E 05 00 04 50:75
8E00:00 0E 15 06 00 00 00 06:31
8E08:01 CC 02 5E 0F 02 59 00:73
8E10:5E 0F 00 06 00 00 00 F5:DA
8E18:FF F5 D5 CD A7 BC D1 F1:2C
8E20:FE 00 21 CD 90 CA DD BC:BA
8E28:ED 53 E3 90 1A 32 E5 90:A3
8E30:3E 01 21 E7 90 CD BC BC:9F
8E38:3E 01 21 F1 90 CD BF BC:E4
8E40:3E 02 21 F8 90 CD BC BC:F5
8E48:3E 02 21 FC 90 CD BF BC:22
8E50:3E 03 21 03 91 CD BC BC:38
8E58:3E 03 21 0A 91 CD BF BC:71
8E60:3E 04 21 11 91 CD BC BC:82
8E68:3E 05 21 1E 91 CD BC BC:C0
8E70:3E 06 21 22 91 CD BC BC:DA
8E78:3E 07 21 2C 91 CD BC BC:0C
8E80:3E 08 21 33 91 CD BC BC:32
8E88:AF 32 E1 90 32 E2 90 32:D2
8E90:E2 8F 3E 01 32 DC 90 2A:3E
8E98:E3 90 23 22 DF 90 CD 66:A0
8EA0:8F 21 CD 90 01 00 81 11:BA
8EA8:AD 8E C3 D7 BC F3 F5 D5:5D
8EB0:E5 C5 DD E5 FD E5 CD C4:E2
8EB8:8E FD E1 DD E1 C1 E1 D1:7F
8EC0:F1 FB ED 4D 3A DC 90 3D:52
8EC8:FE 00 CA D1 8E 32 DC 90:6C
8ED0:C9 06 03 DD 2A DD 90 C5:C8
8ED8:DD E5 11 01 00 DD 7E 00:E4
8EE0:FE 00 CA EB 8E CD 42 8F:50
8EE8:11 05 00 DD E1 C1 DD 19:C3
8EF0:10 E5 DD 7E 00 3C 32 DC:8D
8EF8:90 DD 23 DD 22 DD 90 DD:5D
8F00:7E 00 FE FF C0 AF 32 E2:4B
8F08:8F C3 66 8F 3E 0F FD 21:9D
8F10:07 90 C3 27 8F 3E 0F FD:3B
8F18:21 10 90 C3 27 8F 3E 0F:EB
8F20:FD 21 34 90 C3 27 8F F5:14
8F28:DD E5 FD E5 FD 77 06 CD:36
8F30:B4 8F FD E1 DD E1 F1 3D:22
8F38:FE 09 C2 27 8F FD 36 06:1C
8F40:0F C9 DD 7E 04 FE 10 CA:C7
8F48:0C 8F FE 11 CA 15 8F FE:88
8F50:13 CA 1E 8F FD 21 07 90:84
8F58:3D FE 00 CA B4 8F 11 09:E5
8F60:00 FD 19 C3 58 8F 2A DF:70
8F68:90 23 7E FE 00 C2 80 8F:C3
8F70:AF 32 E1 90 32 E2 90 32:BB
8F78:E2 8F 2A E3 90 23 23 7E:98
8F80:22 DF 90 FE 64 CA EF 8F:48
8F88:FE 65 CA F7 8F DD 2A E5:60
8F90:90 ED 5B E3 90 DD 19 3D:BB
8F98:FE 00 CA A5 8F 11 02 00:56
8FA0:DD 19 C3 97 8F DD 6E 00:DE
8FA8:DD 66 01 ED 5B E3 90 19:68
8FB0:22 DD 90 C9 DD 7E 00 FD:1C
8FB8:77 00 DD 6E 01 DD 66 02:1A
8FC0:CD E7 8F FD 75 03 FD 74:71
8FC8:04 DD 7E 03 FD 77 07 FD:6F
8FD0:36 08 00 FD E5 E1 3A E2:FE
8FD8:90 FE 00 CA AA BC DD 7E:D0
8FE0:00 FE 00 C2 AA BC C9 3A:7C
8FE8:E1 90 FE 00 C8 29 C9 3E:BF
8FF0:01 32 E1 90 C3 66 8F 3E:D3
8FF8:01 32 E2 90 23 7E 32 E2:E3
9000:8F 22 DF 90 C3 66 8F 00:5C
9008:01 00 00 00 00 0F 00 00:F3
9010:00 02 00 00 00 05 0F 00:2B
9018:00 00 03 00 00 00 00 0F:29
9020:00 00 00 04 00 00 00 00:C0
9028:00 00 00 00 02 00 00 00:C2
9030:00 0F 00 00 00 02 00 00:EA
9038:00 0F 0F 00 00 00 02 00:21
9040:00 00 19 0F 00 00 00 05:7F
9048:00 00 00 05 0F 00 00 00:37
9050:05 00 00 00 0F 0F 00 00:8A
9058:00 05 00 00 00 0F 0F 00:B5
9060:00 00 05 00 00 00 00 0F:77
9068:00 00 00 03 01 00 00 00:09
9070:0F 00 00 00 06 00 00 00:2D
9078:01 00 00 00 00 00 02 00:17
9080:00 00 0C 00 00 00 03 02:59
9088:00 00 00 0F 00 00 00 00:54
9090:00 00 00 00 00 00 00 00:20
9098:00 00 00 00 00 00 00 00:28
90A0:00 04 02 00 00 00 00 00:3E
90A8:00 00 00 00 00 00 00 00:38
90B0:00 00 00 03 03 00 00 00:5B
90B8:00 00 00 00 07 00 00 00:6B
90C0:00 00 00 00 00 08 00 00:80
90C8:00 03 00 00 00 00 00 00:5E
90D0:00 00 00 00 00 00 00 00:60
90D8:00 00 00 00 00 00 00 00:68
90E0:00 00 00 00 00 00 00 03:88
90E8:01 00 0A 0A FF 01 01 00:C7
90F0:0A 82 14 01 01 14 FF 01:4C
90F8:01 0F FF 02 82 03 01 03:67
9100:03 FF 03 02 0A FF 08 05:2F
9108:FF 04 03 50 01 01 14 02:90
9110:01 04 01 07 01 03 02 01:F6
9118:01 00 02 0A FF 05 01 0F:70
9120:FF 01 03 01 0A 01 01 00:FE
9128:04 05 FE 01 02 01 0F 05:66
9130:0A FF 02 02 01 0C 01 0C:8B
9138:FF 01 02 02 02 02 02 02:0C
9140:01 01 01 01 01 01 01 01:F5
9148:1C 0A 01 1B 01 01 01 01:92
9150:01 0C 01 01 01 01 1C 01:D8
9158:01 0E 01 01 01 01 01 01:27
9160:01 10 01 01 01 01 01 01:33
9168:01 01 14 01 01 01 01 01:56
9170:01 01 15 16 01 01 01 01:B5
9178:01 01 17 18 01 01 12 01:42
9180:01 01 19 1A 02 02 13 02:72
9188:02 02 01 01 1B 01 01 03:D2
9190:01 1B 0B 01 01 01 01 04:AF
9198:05 01 0D 01 01 1C 01 06:3F
91A0:07 01 0F 1B 01 01 01 08:25
91A8:09 1C 11 01 01 01 01 01:CB
91B0:01 01 01 01 01 01 01 01:65
91B8:01 01 01 01 01 01 01 01:6D
91C0:01 01 01 01 01 01 01 01:75
91C8:01 01 01 1D 01 1D 12 01:0C
91D0:1D 01 02 1E 02 1E 13 02:51
91D8:1E 02 1C 01 01 01 1B 01:B3
91E0:0A 0B 01 01 1B 03 01 01:40
91E8:0C 0D 01 1B 01 04 05 1C:2E
91F0:0E 0F 01 01 01 06 07 01:16
91F8:10 11 1C 01 01 08 09 01:8F
9200:01 1B 01 01 01 01 01 01:EA
9208:01 01 01 01 01 01 01 01:BE
9210:01 01 01 01 01 01 01 01:C6
9218:01 01 01 12 01 01 01 01:12
9220:01 01 02 13 02 02 02 02:3B
9228:02 02 00 00 00 00 00 00:C0
```

Pour un bit de plus

PORT 8 BITS POUR CPC

Le port imprimante du CPC, c'est bien connu, est de type Centronics incomplet. L'ordinateur se contente en effet de fournir 7 bits de données, un signal STB de validation de données et de prendre en compte l'information de disponibilité du terminal qui lui est raccordé.

S'il est impossible, sans changement majeur du soft résident et de l'électronique du CPC, d'augmenter la quantité de données émises, le montage suivant donne la possibilité, moyennant un minimum de matériel, de piloter des systèmes 8 bits à partir des sept disponibles sur le port CPC. Un exemple typique est le pilotage d'une imprimante à code IBM qui exige un code 8 bits. Certaines imprimantes font de même en mode graphique.

Principe

L'octet à émettre est tout d'abord scindé en deux quartets représentant respectivement la partie basse pour les bits de poids faible, et haute pour les bits de poids fort. A chacun de ces quartets, on ajoute une valeur hexa telle que le CPC soit capable de générer le caractère ASCII correspondant. On envoie ensuite ces deux caractères successivement sur le port imprimante, au lieu du seul octet qui serait inévitablement amputé du bit 7. C'est là qu'intervient le petit montage futé...

En réalité, seuls les 4 bits de poids faible des caractères Ascii ci-dessus sont utilisés. Dans un premier temps, est mémorisée dans des bascules la partie basse du caractère correspondant à la partie haute de l'octet de départ, puis on transmet simultanément ces 4 bits et la partie basse du caractère qui représente la partie basse de l'octet et le tour est joué! Comme tout ceci peut paraître confus, un exemple apportera sûrement les éclaircissements nécessaires.

Soit à transmettre l'octet &9C, on peut le représenter comme suit:

D7 D6 D5 D4 D3 D2 D1 D0
1 0 0 1 1 1 0 0

La partie haute est 9, soit 1001, la partie basse est C, soit 1100. Ajoutons à chaque quartet la valeur &40. On obtient &49 et &4C, c'est-à-dire les caractères Ascii «I» et «L». Envoyons vers le montage, «I» puis «L». Celui-ci va prendre la partie basse de «I», soit &9 comme partie haute du caractère à former et la partie basse de «L», soit &C comme partie basse. Il va donc transmettre la valeur &9C, à savoir l'octet de départ. Terminé.

Montage

C'est la simplicité même. Quatre circuits intégrés très courants suffisent pour obtenir le résultat escompté. A la mise sous tension, ou lors d'un Reset, la bascule «D» a ses sorties respectivement à 0 pour Q et 1 pour /Q. Lors de l'envoi du premier caractère («I» dans notre exemple), la partie basse est mémorisée dans le latch 74LS373 sur le front de descente de /STB. Pendant toute la durée de ce premier /STB, la bascule «D» 74LS74 n'autorise son passage que vers le 74LS373. Quand le /STB remonte, la bascule «D» change d'état. Donc, le deuxième /STB passera vers la sortie alors qu'il n'atteindra pas le latch. Lorsque le caractère «L» de l'exemple sera envoyé, les 4 bits de poids faible seront aiguillés vers les poids faibles de la sortie et les données mémorisées dans le latch seront envoyées vers les poids forts.

Le front de remontée du signal /STB sera pris en compte par la bascule «D» qui reviendra de ce fait à son état initial, aiguillant à nouveau le /STB vers le latch.

Une précaution indispensable, le circuit «R-C» sur la RAZ de la bascule «D», est agrémenté d'un poussoir de Reset, ce qui permet de démarrer à tous les coups dans le bon sens et de réinitialiser si nécessaire le montage.

L'inconvénient majeur inhérent à ce montage est le ralentissemnt de la transmission. Il faut en effet exécuter deux PRINT au lieu d'un et faire en outre une petite conversion pour fabriquer les caractères. L'utilisation du port 8 bits est toutefois d'une simplicité enfantine: il suffit de remplacer les PRINT #8 par un GOSUB appelant la ligne Basic suivante:

PRINT #8, CHR$(INT (b/16) + 64); CHR$ (b MOD 16 + 64) ;;
RETURN

où «b» est la valeur à convertir qui peut être exprimée en hexadécimal ou en décimal. On peut bien sûr remplacer «b» par n'importe quelle variable numérique. Bien entendu, il est facile, sur le

même principe, d'envoyer des caractères Ascii nouveaux par le sous-programme suivant, où b$ est un caractère Ascii:
PRINT #8, CHR$(INT (ASC (b$)/16 + 64)); CHR$ (ASC(b$) MOD 16 + 64)
Ceci afin de ne pas débrancher le montage.

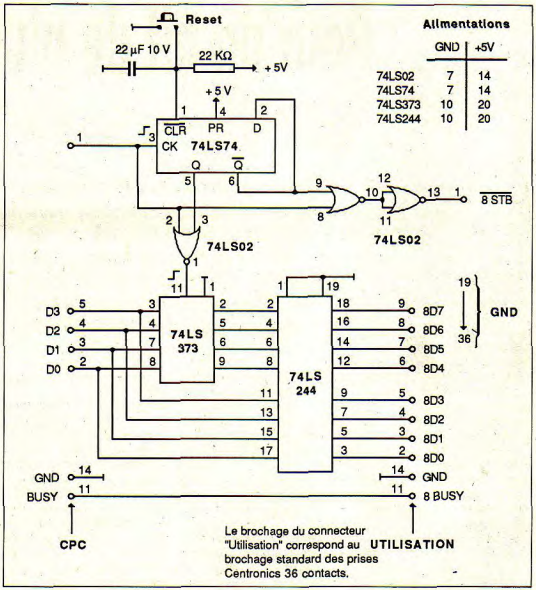
Mise au point

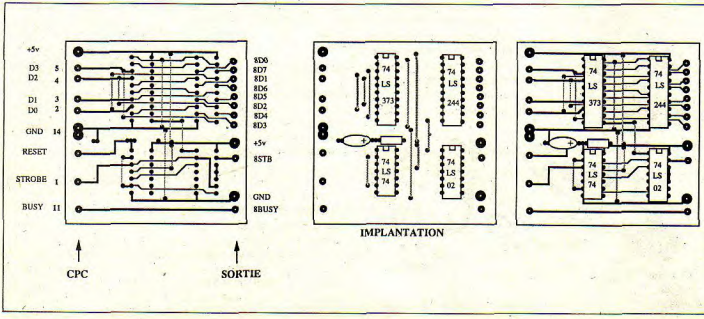
Il y a vraiment peu de chose à en dire. Si le câblage est réalisé convenablement, le circuit démarrera du premier coup. L'éternel testeur à DEL (*Micro-Mag* n°2, *«Les liaisons dangereuses»*) est encore d'actualité.

- A la mise sous tension, vérifier la présence d'un 1 en 6 du LS74, et d'un 0 en 5.
- Vérifier l'efficacité du bouton Reset en 1 du même boîtier.
- Envoyer par un PRINT #8, un caractère (un seul). Contrôler la présence de sa partie basse sur les sorties du 74LS373.
- Recommencer éventuellement avec d'autres caractères (après un Reset pour chaque nouveau caractère).

Parvenu à ce point, les chances sont grandes pour que le montage soit prêt à vous rendre les services attendus.

Michel Hugot





Mettez les pouces

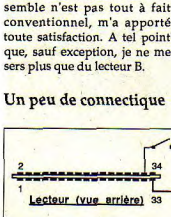
LECTEUR 5 POUCES 1/4 POUR CPC 6128

Souvent confronté à une valse incessante entre deux disquettes, qui n'a rêvé d'acquérir à moindre frais un second lecteur?

Par ailleurs, le prix de la disquette 3 pouces est encore élevé et ce standard étant l'exclusivité d'Amstrad, ce n'est pas demain qu'on pourra s'en procurer par centaines. Enfin, les disquettes 5 pouces 1/4 (360 Ko) coûtent actuellement dans le commerce moins de 2,50F, ce qui milite sérieusement en leur faveur.

Devant ces constatations, j'ai saisi l'opportunité d'un lecteur 5 pouces 1/4 à prix intéressant (environ 600F). Moyennant quelques minutes de réflexion devant les «docs», un câble fut vite confectionné pour le relier à mon CPC chéri... Je vous livre ici le résultat qui, même si le fonctionnement de l'ensemble n'est pas tout à fait conventionnel, m'a apporté toute satisfaction. A tel point que, sauf exception, je ne me sers plus que du lecteur B.

Un peu de connectique



Le connecteur pour lecteur de disquettes supplémentaire du CPC semble présenter un brochage standard. Mais ne vous y fiez pas, les signaux qui y aboutissent ont été inversés (1 ->34, 2 ->33, etc.). Consultez plutôt le tableau ci-contre:

Malheureusement, le signal qui devrait permettre de choisir la face de la disquette que l'on veut accéder est forcé à «0» en permanence par le CPC. Or, on ne peut retourner les disquettes 5 pouces 1/4 comme les disquettes 3 pouces. Rassurez-vous, une solution existe, plus rapide qu'un retournement. En effet, il suffit de ne pas effectuer la connection 3 CPC ->32 lecteur et de monter un interrupteur entre 32 du lecteur et la masse. En position ouverte, l'interrupteur laissera la broche 32 à «1», alors qu'en position fermée, il forcera cette broche à «0». Cela permet d'accéder les deux faces par un simple basculement. Un conseil: pour éviter les hésitations, il est pratique de positionner l'interrupteur de façon à enregistrer la face supérieure quand il est basculé vers le haut. On s'habitue vite à ce genre de petit détail confortable...

CPC	Nom du signal	Lecteur
1	_Ready	34
3	_Side 1 select	32
5	_Read data	30
7	_Write protect	28
9	_Track 0	26
11	_Write gate	24
13	_Write data	22
15	_Step	20
17	_Direction select	18
19	_Motor ON	16
21	(non utilisé sur CPC)	14
23	_Drive select 1	12
25	(non utilisé sur CPC)	10
27	_Index	8
29	(non utilisé sur CPC)	6
31	(non utilisé sur CPC)	4
33	(non utilisé sur CPC)	2
Bornes impaires	Masse	Bornes paires

Les signaux précédés du signe « _ » sont actifs bas.

Le matériel

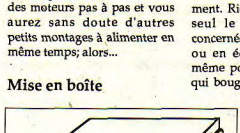
Les connecteurs nécessaires sont de type «encartable» 34 contacts. Ils sont très courants chez les revendeurs spécialisés, de même que le connecteur d'alimentation.

N'importe quel type d'interrupteur convient.

L'alimentation

Optez pour le confort, choisissez une alimentation capable de fournir 300 mA sur le 5v et 500 mA sur le 12v (celle proposée dans ce magazine). C'est trop, mais cela vous mettra à l'abri des parasites induits de l'alim. par les commutations des moteurs pas à pas et vous aurez sans doute d'autres petits montages à alimenter en même temps; alors...

Mise en boîte



Aspect du lecteur en boîte

Aspect à ne pas négliger pour un lecteur de disquettes si on veut qu'il dure longtemps. Ce travail est très simple à réaliser, le chassis du lecteur comportant des trous taraudés tout à fait adaptés.

Une tôle d'aluminium dessous, avec des pieds caoutchoutés pour le confort et un «U» en alu décoré pour cacher la mécanique, fournissant du même coup un support d'interrupteur et le tour est joué!

Remarques sur le fonctionnement

La gestion des lecteurs de disquettes du CPC correspond à un «système minimum», c'est-à-dire que ne sont générés que les signaux indispensables. Nous avons déjà vu qu'une seule face était explorée, mais par ailleurs, la commande Motor ON est unique. Cela implique que les moteurs des deux lecteurs seront sollicités simultanément. Rien de grave, puisque seul le lecteur réellement concerné sera accédé en lecture ou en écriture. Il en va de même pour le bras porte tête qui bougera au même rythme sur les deux lecteurs sans conséquence grave. Si le lecteur externe est raccordé, il est indispensable de l'alimenter, sous peine de voir le lecteur interne tourner sans cesse, le rendant indisponible.

Michel Hugot

Les modes d'adressage

AUTO-FORMEZ-VOUS A L'ASSEMBLEUR 68000

(4e partie)

Après cette longue attente, douloureuse pour les plus assidus d'entre-vous, notre initiation se poursuit avec les modes d'adressage du 68000.

Précisons toutefois que nous n'en sommes pas encore à la description détaillée des instructions du 68000.. Pour l'heure, nous allons essayer de mettre en oeuvre la programmation Assembleur à travers une unique instruction, qui, à elle seule, réalise la majorité des transferts de données et d'adresses. De plus, les modes d'adressage seront les mêmes pour toutes les instructions. Ci-contre, une vue d'ensemble des 12 modes d'adressage que nous allons étudier.

L'instruction MOVE (transfert)

C'est bien simple, dans un programme, la moitié des instructions rencontrées sont des MOVEs. Cette instruction n'est certes pas la seule à effectuer des transferts en mémoire, mais le nombre de combinaisons différentes quelle permet la rend la plus importante et la plus caractéristique des MC680xx.

La tâche d'une instruction MOVE est le transfert de données d'un endroit vers un autre. Lesdits «endroits» peuvent être, soit la mémoire interne du 68000 (registres internes), soit une mémoire externe à celui-ci (RAM, ROM, PROM, EPROM, EEROM). RAM et ROM étant utilisées dans l'immense majorité des cas, nous passerons les autres sous silence (ces articles se bornent en effet à une étude pratique du 68000). A noter que les transferts sont de type I/O (In/Out), c'est-à-dire dans les 2 sens, sans risque de conflit électrique.

Instruction MOVE générale

MOVE(+format) opérande source, opérande destination

Les opérandes peuvent être: un registre de données, un registre d'adresses, une adresse absolue, une adresse relative, un déplacement codé sur 8 ou 16 bits par rapport au PC, une valeur immédiate, etc. L'instruction MOVE consiste en une opération de lecture (opérande source) et une d'écriture (opérande destination). La seule chose à retenir ici est l'ordre des opérandes dans l'instruction.

Type d'adressage	Symbole
Direct de registre de données	Dn
Direct de registres d'adresses	An
Absolu long	xxxx.L
Absolu court	xxxx.W
Immédiat	#xxxx
Indirect de registres d'adresses	(An)
Indirect de registres d'adresses avec postincrémentation	(An)+
Indirect de registres d'adresses avec prédécrémentation	-(An)
Indirect de registres d'adresses avec déplacement codé sur 16 bits	d16(An)
Indirect de registres d'adresses indexés avec déplacement sur 8 bits	d8(An,Rn)
Relatif au PC avec déplacement codé sur 16 bits	d16(PC)
Relatif au PC indexé avec déplacement codé sur 8 bits	d8(PC,Rn)

Dn : un des 8 registres de données suivants : D0, D1, D2, D3, D4, D5, D6, D7
An : un des 8 registres d'adresses suivants : A0, A1, A2, A3, A4, A5, A6, A7
xxxx : donnée codée sur 16 ou 32 bits
d16 : déplacement sur 16 bits
d8 : déplacement sur 8 bits
Rn : lire : Dn ou An

Les 12 modes d'adressage du 68000

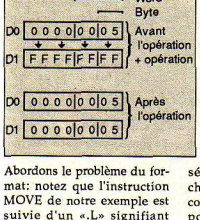
Adressage direct de registres de données

Comme son nom l'indique, ce premier mode d'adressage permet de transférer «directement» le contenu d'un registre de données vers un

autre sans passer par la mémoire externe.

Formulation générale:
MOVE.format Dn, Dn
Opération effectuée:
Dn -> Dn
Soit un exemple concret:
MOVE.L D0, D1

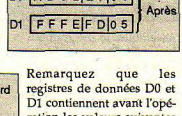
Voyez la figure ci-dessous en considérant que le registre de donnees D0 contient au départ la valeur 5 et surtout en ne vous demandant pas comment on en est arrivé là (si vous saviez...)
Explications: le contenu du registre de données D0 qui représente l'opérande source, est transféré dans D1 qui officie pour l'opérande destination.



Abordons le problème du format: notez que l'instruction MOVE de notre exemple est suivie d'un «.L» signifiant Long Word, soit 32 bits. De ce fait, l'intégralité du contenu de D0 sera transférée. Nous savons en effet depuis l'article précédent (les registres internes) qu'un registre de données contient 32 bits. Nous sommes également au fait de deux autre formats :

- B : Byte (8 bits)
- W : Word (16 bits)

Soit au total 3 formats différents, d'ailleurs valables pour tous les autres modes d'adressage. A dire vrai, leur présence facilite considérablement la tâche du programmeur, car ils indiquent la partie du registre de données qui devra être transférée et le cas échéant modifiée. Si le format est B, seuls les 8 bits (les 8 de plus faible poids) du registre de données seront pris en compte. Si en revanche le format est W, ne seront considérés que les 16 bits de plus faible poids (bit 0-15). Comme d'habitude, un exemple appuyé par une figure : MOVE.B D0, D1



Remarquez que les registres de données D0 et D1 contiennent avant l'opération les valeurs suivantes (respectivement): $ABCDEF05 et $FFFEFD41. Nous verrons, dans un prochain mode d'adressage, comment modifier directement le contenu de Dn (voir Adressage immédiat).
Déjà, nous percevons la puissance du 68000 : les instructions sont optimisées au maximum et pour chaque cas, on trouve celle correspondante. En outre, du point de vue de la place mémoire, le gain se révèle énorme même pour un nombre restreint d'instructions. Reste un problème: leur syntaxe . Dorénavant, «Dn» signifiera pour vous un des 8 registres de données (D0-D7), quel qu'il soit. D'autre part, la syntaxe correcte avec ce mode d'adressage est :

MOVE. (<-point) format (B, W, L) [espace] Dn , (virgule) Dn

Soit d'une façon plus lisible MOVE.f Dn, Dn, à l'exclusion de tout autre caractère. Si le format utilisé est W, ce dernier peut être omis lors de la saisie d'un prg. assembleur (surtout ne vous en privez pas!), ainsi, MOVE.W D0, D5 devient MOVE D0, D5. '
Enfin, le manque d'intérêt d'une instruction du genre : MOVE.L D0, D0 est évident. Bien que correcte elle ne sert à rien.

Adressage direct de registres d'adresses

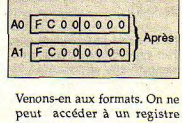
Nous avons vu comment adresser un registre de données, occupons-nous maintenant des registres d'adresses.

Formulation Générale:
MOVEA.f An, An
Opération effectuée:
An -> An

Observez la lettre A accolée au MOVE. Cette dénomination est obligatoire lorsque l'opérande destination est un registre d'adresses. Sachez néanmoins que le code opératoire en langage machine résultant de MOVEA.f An, An est strictement identique à celui généré par MOVE.f An, An. Les programmeurs qui ont réalisé les moniteurs assembleurs sur nos 16 bits s'en sont rendu compte, c'est pourquoi la plupart d'entre-eux négligent cet ajout. Il en résulte qu'on peut taper MOVE à la place de MOVEA. Lors du listage correspondant (désassemblage), on obtiendra soit MOVE, soit MOVEA. Soyez rassurés, il n'y a aucune différence. Certains assembleurs sont déjà rompus à cette méthode; ainsi le K-Seka n'accepte pas les MOVEA mais seulement les MOVE. Dans cette initiation, nous utiliserons d'abord MOVEA, mais cela ne durera pas. A mon sens, cette particularité ne fait qu'allourdir la difficulté de compréhension.

Il y a finalement très peu de nouveautés par rapport au mode d'adressage précédent. Le contenu d'un registre d'adresses est transféré dans un autre, «A» signifiant l'un des 8 registres d'adresse, que nous avons passés en revue lors d'un précédent article (A0-A7). On peut donc, à priori, manipuler le registre de la pile, soit A7. Avec une instruction du genre MOVEA.L A7, A0 on récupère la pile puis on stocke son pointeur dans A0. Si en revanche on tape MOVEA.L A0, A7, cela signifiera que le pointeur de pile va changer de valeur et contenir ce que contient A0. Il faut par conséquent être très prudent avec cette instruction, car le SP (Stack Pointer) modifié lors de l'exécution d'un sous-programme, rend impossible le retour au programme principal, l'adresse de retour n'étant plus pointée par SP. Pour l'instant, le peu que nous savons ne nous permet pas d'écrire un programme d'exemple convaincant. A propos de la pile, nous vous en offrirons un bientôt.

Revenons à nos moutons, voici un exemple : MOVEA.L A0, A1 sachant que, avant l'opération, A0 valait $FC000000 et A1 $12345678.

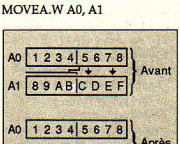


Venons-en aux formats. On ne peut accéder à un registre d'adresses avec un format du type B. Nous sommes donc obligés de transférer soit 16 bits soit 32 bits, cette dernière variante utilisant l'intégralité d'un registre d'adresses.

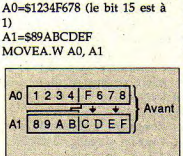
Occupons-nous du format W (16 bits) en faisant appel aux nombres signés. En effet, le bit 15 du registre d'adresse source déterminera le contenu des 16 bits de plus fort poids du registre destination. Il y donc 2 possibilités, voyons-les en

détails à l'aide d'exemples :

• **1er cas**
A0=$12345678 (le bit 15 est à 0)
A1=$89ABCDEF
MOVEA.W A0, A1



• **2e cas**
A0=$1234F678 (le bit 15 est à 1)
A1=$89ABCDEF
MOVEA.W A0, A1



Résumons-donc la situation :
- bit 15 de A0 à 1 -> bits 16-31 de A1 à 1
- bit 15 de A0 à 0 -> bits 16-31 de A1 à 0

Nous avons procédé à ce que l'on appelle une extension de signe 32 bits, en amenant à 32 bits un nombre de «n» bits. Dans notre cas, n signifie 16 mais nous verrons une autre variante prochainement avec n=8 (Dn).

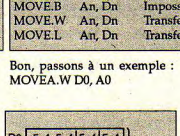
• **Règle pratique:** les extensions de signes ne sont effectuées qu'avec des registres d'adresses et donc seulement lorsque le format est le mot (16 bits). D'autre part, seul l'opérande destination est affecté. Il est possible de réaliser une extension de signe avec un registre de données mais cela ne se produit que lorsque c'est implicitement demandé par le programmeur, c'est-à-dire en utilisant l'instruction MOVEQ. Il n'est pas possible d'obtenir une extension de signe avec un adressage direct de registres de données. Cependant, lors d'un adressage direct mixte du genre : MOVEA.W D0, A0, il y a une extension.

Adressage direct mixte

Il ne s'agit pas à proprement parler d'un nouveau mode d'adressage, puiqu'il intègre les deux que nous venons d'étudier. En effet, celui-ci permet de transférer le contenu d'un registre de données vers un registre d'adresses, et vice-versa bien évidemment. Les particularités dues aux formats et aux extensions sont telles qu'un tableau est nécessaire.

Instruction	Opération
MOVEA.B Dn, An	Impossible! Format B interdit.
MOVEA.W Dn, An	Extension de signe 32 bits avant transfert.
MOVEA.L Dn, An	Transfert des 32 bits, classique.
MOVE.B An, Dn	Impossible! Format B interdit.
MOVE.W An, Dn	Transfert des 16 bits de poids faible, pas d'extension.
MOVE.L An, Dn	Transfert 32 bits classique.

Bon, passons à un exemple : MOVEA.W D0, A0



Le moniteur assembleur

Il est temps de lancer votre moniteur préféré (de préférence ceux cités dans le 1er article). Je vais vous indiquer la marche à suivre pour tester vous même des MOVEs, ceci à l'aide d'un mini-programme.
Passez en mode édition d'un programme source en vous reportant au manuel d'utilisation du logiciel (n'espérez pas trouver dans ces articles, la documentation et les diverses explications nécessaires à l'emploi du K-Seka, Profimat, Devpac, etc. Pour les utiliser, achetez l'original! L'investissement de départ sera largement récupéré quand vous serez à même de produire vos propres programmes). Tapez le programme suivant :

```
start:                  ; Label de départ
MOVE.L #$12345678, D0   ; D0=$12345678
MOVEA.L #$89ABCDEF, A0  ; A0=$89ABCDEF
MOVE.W A0, D0           ; A0 -> D0 (sur 16 bits)
RTS                     ; retour à l'éditeur.
```

Assemblez le programme et lancez-le. En retour, l'éditeur vous fournira la valeur de chacun des registres du 68000, et notamment, dans notre programme gramme de test, la nouvelle valeur de D0 ($1234CDEF). Pas de précisions supplémentaires sur ce programme qui utilise l'adressage immédiat et est suffisamment clair et concis pour vous permettre de tester vos connaissances fraîchement acquises. Amusez-vous avec, changez les MOVE, testez, etc. Il n'y a pas de meilleur moyen de progresser.

• **A propos de la syntaxe:** Le «;» correspond au début d'un commentaire (équivalent du REM Basic). On peut mettre autant d'espaces que l'on veut entre l'instruction même et l'opérande source et entre l'opérande destination et l'éventuel commentaire.

A titre de conclusion, une remarque intéressante (enfin!): le format B est interdit lors de l'utilisation de l'adressage direct de registres d'adresses ou encore d'un adressage direct mixte. En premier lieu, il n'y a rien à craindre, car l'éditeur assembleur se charge de vous le faire remarquer en stoppant l'assemblage en cours pour vous permettre de réparer la bévue. En revanche, on peut "forcer" un MOVE.B en tapant directement les codes opératoires correspondants directement dans la RAM. Après lancement du programme contenant ladite instruction et lorsque le PC atteint notre MOVE, le 68000 déclenche une exception «Instruction Illégale». C'est-à-dire qu'un programme va être lancé sans même notre autorisation. De plus, celui-ci aura pour but principal de nous rappeler qu'il y a un bug.

La conséquence de ceci est que la machine sera ensuite plantée et relancée, ce qui se traduit dans notre jargon par une activation de la broche RESET du 68000. Je ne résiste pas au plaisir de vous révéler que le lancement des exceptions (sortes d'interruptions) est vectorisé. Il est par conséquent tout à fait possible de tout détourner et donc de prendre le contrôle total de notre machine (cela me semble normal et justifié!). Ce principe est utilisé pour les protections des jeux à un niveau supérieur, bien sûr, mais chuuuut!, l'A.P.P. nous surveille!..

Bientôt la suite de notre grande saga.

Stéphane Rodriguez

Hooou, les cornes!

PACO

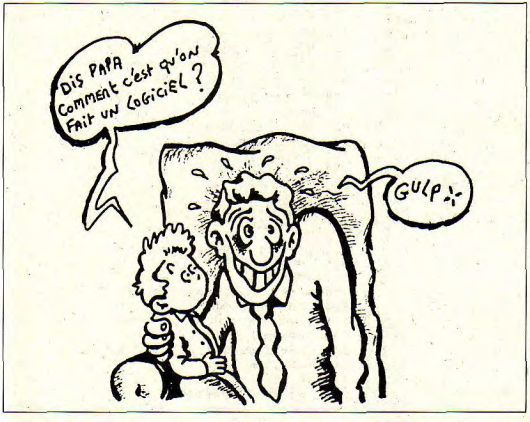
Seul espoir de Paco, affublé d'une paire de cornes à la suite d'un sortilège, engloutir les perles d'énergie du temple de la beauté.

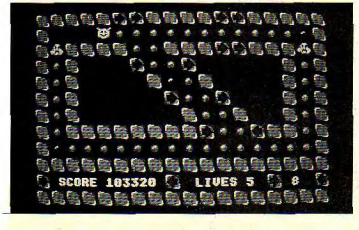
Cet astucieux Pac-Man en GFA réclame plus que tout autre réflexe et stratégie. Les fantômes rapides comme l'éclair ne laissent en effet aucune chance au héros, s'il n'a pris la peine d'étudier préalablement (et rapidement) le décor afin de déterminer le meilleur trajet. Dix tableaux sont ainsi offerts à la perspicacité du joueur.

- Les pastilles bleues stoppent les fantômes pendant un laps de temps.
- Les manœuvres de directions doivent être rapides et précises.
- La sélection du joystick se fait par appui sur Fire et celle du clavier (touches directionnelles + barre d'espacement) par la barre d'espacement.
- L'abandon, selon la sélection choisie, s'obtient par Fire ou la barre d'espacement.

Ne soyez pas surpris de l'apparence de ce listing faisant appel au vérificateur GFA V.1.0 (reportez-vous à son mode d'emploi). Les lecteurs infaillibles peuvent cependant taper normalement les lignes sans tenir compte des numéros et des sommes de contrôle.

Luc, Hervé & Claude Guillaume





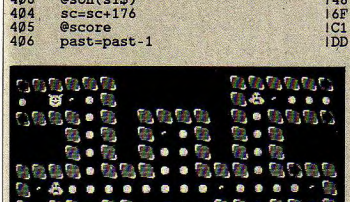

```
1  '  ************************          |3Ø
2  '  **                    **          |CA
3  '  *        P A C O       *          |48
4  '  *                      *          |76
5  '  *   Luc, Herve and Claude *       |72
6  '  *       Guillaume      *          |B8
7  '  *    BLACK SYSTEM 1989  *         |DF
8  '  **                    **          |CA
9  '  ************************          |3Ø
1Ø '                                     |27
11 debut:                                |CD
12 RESERVE 65535                         |C9
13 ON BREAK CONT                         |C4
```

```
14 VOID XBIOS(5,L:-1,L:-1,Ø)                |7Ø
15 FOR t=Ø TO 15                            |CE
16   SETCOLOR t,Ø                           |DF
17 NEXT t                                   |8B
18 IF PEEK(15Ø0ØØ)=96                       |CØ
19   GOTO debut1                            |4F
2Ø ENDIF                                    |32
21 RESTORE routine                          |9F
22 @pokage(15ØØØØ,84)                       |11
23 RESTORE dessins                          |6D
24 @pokage(48ØØØØ,784)                      |5Ø
25 debut1:                                  |2E
26 SPOKE &H484,14                           |2Ø
27 HIDEM                                    |3E
28 RESTORE son1                             |FØ
29 FOR t=1 TO 17                            |EF
3Ø   READ v                                 |DD
31   s1$=s1$+CHR$(v)                        |21
32 NEXT t                                   |8B
33 RESTORE son2                             |FC
34 FOR t=1 TO 17                            |EF
35   READ v                                 |DD
36   s2$=s2$+CHR$(v)                        |2D
37 NEXT t                                   |8B
38 @mess(6Ø,15,14,15,"Claude Guilla
   ume presents")                           |F2
39 @mess(51,155,14,15,"Program ....
   .. Luc Guillaume")                       |ØC
4Ø @mess(51,175,14,15,"Graphics ...
   Herve Guillaume")                        |1F
41 @mess(99,195,14,15,"BLACK SYSTEM
   1989")                                   |F3
42 @mess(67,135,3,11,"Press FIRE to
   start game")                             |16
43 RESTORE pres                             |A8
44 ad=XBIOS(2)+4336                         |FD
45 FOR y=1 TO 6                             |78
46   esp=Ø                                  |4E
47   FOR x=1 TO 16                          |C5
48     READ b                               |5A
49     IF b=1                               |57
5Ø       @affsp(ad+esp,48ØØØØ,8,14)         |13
51     ENDIF                                |ØD
52     IF b=2                               |61
53       @affsp(ad+esp,48Ø336,8,14)         |42
54     ENDIF                                |ØD
55     IF b=3                               |6B
56       @affsp(ad+esp,48Ø448,8,14)         |A7
57     ENDIF                                |ØD
58     ADD esp,8                            |C6
59   NEXT x                                 |B3
6Ø   ADD ad,2400                            |85
61 NEXT y                                   |A9
62 RESTORE coul                             |71
63 FOR t=Ø TO 15                            |CE
64   READ coul                              |65
65   SETCOLOR t,coul                        |A9
66 NEXT t                                   |8B
67 fire=XBIOS(34)+5Ø                        |91
68 clavier=XBIOS(34)+1Ø9                    |B1
69 testfire:                                |33
7Ø IF PEEK(fire)=249                        |7C
71   @seljoy                                |13
72   GOTO start                             |A5
73 ENDIF                                    |32
74 IF PEEK(clavier)=57                      |25
75   @selkeys                               |49
76   GOTO start                             |A5
77 ENDIF                                    |32
78 GOTO testfire                            |E6
79 start:                                   |EB
8Ø GRAPHMODE 1                              |3E
81 DIM br(18,1Ø),xx(2),yy(2),fan(2)         |ØE
82 RESTORE tab1                             |E9
83 sc=Ø                                     |B2
84 ntab=Ø                                   |57
85 vie=9                                    |8C
86 start1:                                  |4C
87 @scroll                                  |C4
88 WAVE Ø,Ø                                 |69
89 ch=Ø                                     |AC
9Ø DEFTEXT 15,Ø,Ø,6                         |72
91 GOSUB tableaux                           |8A
92 GOSUB score                              |34
93 x=2                                      |89
94 y=2                                      |8A
95 pac=XBIOS(2)+4Ø16                        |24
96 clavier:                                 |96
97 PAUSE 2                                  |99
98 IF PEEK(adfire)=fi                       |17
99   GOTO abort                             |FF
1ØØ ENDIF                                   |32
1Ø1 IF PEEK(adkeys)=dr                      |BD
1Ø2   @droite                               |97
1Ø3 ENDIF                                   |32
1Ø4 IF PEEK(adkeys)=ga                      |BE
1Ø5   @gauche                               |E8
1Ø6 ENDIF                                   |32
1Ø7 IF PEEK(adkeys)=ha                      |CF
1Ø8   @haut                                 |94
1Ø9 ENDIF                                   |32
11Ø IF PEEK(adkeys)=ba                      |69
111   @bas                                  |43
112 ENDIF                                   |32
113 @affsp(pac,48Ø112,8,14)                 |89
114 FOR num=1 TO 2                          |D2
115   @depfany                              |BC
116   @depfanx                              |B2
117   @affsp(fan(num),48Ø672,8,14)          |25
118 NEXT num                                |Ø4
119 IF ch=1                                 |93
12Ø   GOTO changement                       |8C
121 ENDIF                                   |32
122 IF stop<>Ø                              |E5
123   GOTO clavier1                         |C4
124 ENDIF                                   |32
125 IF xx(1)=x AND yy(1)=y                  |86
126   GOTO mort                             |97
127 ENDIF                                   |32
128 IF xx(2)=x AND yy(2)=y                  |AØ
129   GOTO mort                             |97
13Ø ENDIF                                   |32
131 clavier1:                               |59
132 IF stop<>Ø                              |E5
133   SUB stop,1                            |FD
134 ENDIF                                   |32
135 GOTO clavier                            |Ø3
136 mort:                                   |97
```

```
137 @son(s2$)                          !9B
138 SUB vie,1                          !EA
139 IF vie=-1                          !4B
140   GOTO abort                       !FF
141 ENDIF                              !32
142 @vie                               !FE
143 PAUSE 50                          !2F
144 IF ntab=1                          !CD
145   RESTORE tab1                     !A3
146 ENDIF                              !32
147 IF ntab=2                          !D6
148   RESTORE tab2                     !B1
149 ENDIF                              !32
150 IF ntab=3                          !DF
151   RESTORE tab3                     !BF
152 ENDIF                              !32
153 IF ntab=4                          !E8
154   RESTORE tab4                     !CD
155 ENDIF                              !32
156 IF ntab=5                          !F1
157   RESTORE tab5                     !DB
158 ENDIF                              !32
159 IF ntab=6                          !FA
160   RESTORE tab6                     !E9
161 ENDIF                              !32
162 IF ntab=7                          !03
163   RESTORE tab7                     !F7
164 ENDIF                              !32
165 IF ntab=8                          !0C
166   RESTORE tab8                     !05
167 ENDIF                              !32
168 IF ntab=9                          !15
169   RESTORE tab9                     !13
170 ENDIF                              !32
171 IF ntab=10                         !AE
172   RESTORE tab10                    !74
173 ENDIF                              !32
174 SUB ntab,1                         !56
175 GOTO start1                        !50
176 PROCEDURE depfanx                  !50
177   IF stop>1                        !C8
178     GOTO depfanx1                  !9D
179   ENDIF                            !60
180   IF xx(num)>x                     !17
181     @fanga                         !6C
182     GOTO depfanx1                  !9D
183   ENDIF                            !60
184   IF xx(num)<x                     !FD
185     @fandr                         !FB
186     GOTO depfanx1                  !9D
187   ENDIF                            !60
188   depfanx1:                        !74
189 RETURN                             !9E
190 PROCEDURE depfany                  !61
191   IF stop>1                        !C8
192     GOTO depfany1                  !AD
193   ENDIF                            !60
194   IF yy(num)>y                     !32
195     @fanha                         !75
196     GOTO depfany1                  !AD
197   ENDIF                            !60
198   IF yy(num)<y                     !18
199     @fanba                         !3F
200     GOTO depfany1                  !AD
201   ENDIF                            !60
202   depfany1:                        !7D
203 RETURN                             !9E
204 PROCEDURE fandr                    !25
205   IF br(xx(num)+1,yy(num))=1       !B1
206     GOTO fandr1                    !EB
207   ENDIF                            !60
208   @affel(fan(num),490000,8,14)     !62
209   xx(num)=xx(num)+1                !64
210   ADD fan(num),8                   !A6
211   fandr1:                          !4B
212 RETURN                             !9E
213 PROCEDURE fanga                    !50
214   IF br(xx(num)-1,yy(num))=1       !D1
215     GOTO fanga1                    !24
216   ENDIF                            !60
217   @affel(fan(num),490000,8,14)     !62
218   xx(num)=xx(num)-1                !88
219   SUB fan(num),8                   !17
220   fanga1:                          !E6
221 RETURN                             !9E
222 PROCEDURE fanha                    !5E
223   IF br(xx(num),yy(num)-1)=1       !43
224     GOTO fanha1                    !31
225   ENDIF                            !60
226   @affel(fan(num),490000,8,14)     !62
227   yy(num)=yy(num)-1                !A6
228   SUB fan(num),2400                !1E
229   fanha1:                          !EC
230 RETURN                             !9E
231 PROCEDURE fanba                    !0A
232   IF br(xx(num),yy(num)+1)=1       !13
233     GOTO fanba1                    !E3
234   ENDIF                            !60
235   @affel(fan(num),490000,8,14)     !62
236   yy(num)=yy(num)+1                !82
237   ADD fan(num),2400                !AD
238   fanba1:                          !C8
239 RETURN                             !9E
240 PROCEDURE pokage(ad,taille)        !81
241   FOR l=1 TO taille/14             !A3
242     FOR t=0 TO 13 STEP 2           !CC
243       READ val$                    !DB
244       DPOKE ad,VAL("&"+val$)       !93
245       ad=ad+2                      !39
246     NEXT t                         !1B
247   NEXT l                           !53
248 RETURN                             !9E
249 abort:                             !DF
250 @affsp(XBIOS(2)+11256,481000,48,
    44)                                !30
251 @mess(124,97,14,15,"GAME OVER")    !46
252 @son(s2$)                          !9B
253 PAUSE 250                          !F3
254 SOUND 0,0,0,0                      !EF
255 CLEAR                              !4B
256 @scroll                            !C4
257 GOTO debut                         !94
258 gagne:                             !70
259 @affsp(XBIOS(2)+11248,481000,64,
    44)                                !06
260 @mess(100,97,14,15,"Congratulati
    ons")                              !02
261 @son(s2$)                          !9B
```

```
262 PAUSE 250                          !F3
263 SOUND 0,0,0,0                      !EF
264 CLEAR                              !4B
265 @scroll                            !C4
266 GOTO debut                         !94
267 changement:                        !80
268 ch=0                               !AC
269 PAUSE 10                          !13
270 FOR t=50 TO 200 STEP 10            !4B
271   SOUND 1,14,#t,1                  !EC
272   SOUND 0,0,0,0                    !2B
273   sc=sc+879                        !CF
274   @score                           !C1
275 NEXT t                             !8B
276 PAUSE 25                          !42
277 IF ntab=10                         !AE
278   GOTO gagne                       !F6
279 ENDIF                              !32
280 @scroll                            !C4
281 @tableaux                          !DC
282 x=2                               !89
283 y=2                               !8A
284 pac=XBIOS(2)+4016                  !24
285 GOTO clavier                       !03
286 PROCEDURE selkeys                  !9E
287   fi=57                            !2B
288   dr=77                            !55
289   ga=75                            !0C
290   ha=72                            !FA
291   ba=80                            !E0
292   adfire=XBIOS(34)+109             !FE
293   adkeys=XBIOS(34)+109             !A0
294 RETURN                             !9E
295 PROCEDURE seljoy                   !82
296   fi=249                           !CD
297   dr=8                             !D9
298   ga=4                             !86
299   ha=1                             !77
300   ba=2                             !6B
301   adfire=XBIOS(34)+50              !67
302   adkeys=XBIOS(34)+61              !32
303 RETURN                             !9E
304 PROCEDURE droite                   !9D
305   IF x=18                          !F4
306     @affsp(pac,490000,8,14)        !23
307     x=1                            !61
308     y=2                            !6D
309     pac=XBIOS(2)+4008              !38
310     @testbord                      !B6
311     GOTO droite1                   !18
312   ENDIF                            !60
313   IF br(x+1,y)=1                   !C1
314     GOTO droite1                   !18
315   ENDIF                            !60
316   IF br(x+1,y)=0                   !B1
317     br(x+1,y)=5                    !3D
318     @points                        !D0
319   ENDIF                            !60
320   IF br(x+1,y)=2                   !D1
321     br(x+1,y)=5                    !3D
322     sc=sc+500                      !3E
323     stop=35                        !32
324     @points                        !D0
325   ENDIF                            !60
326   @affsp(pac,490000,8,14)          !FC
327   x=x+1                            !72
328   ADD pac,8                        !03
329   droite1:                         !4E
330 RETURN                             !9E
331 PROCEDURE gauche                   !38
332   IF x=1                           !FB
333     @affsp(pac,490000,8,14)        !23
334     x=18                           !22
335     y=2                            !6D
336     pac=XBIOS(2)+4144              !47
337     @testbord                      !B6
338     GOTO gauche1                   !CD
339   ENDIF                            !60
340   IF br(x-1,y)=1                   !D5
341     GOTO gauche1                   !CD
342   ENDIF                            !60
343   IF br(x-1,y)=0                   !C5
344     br(x-1,y)=5                    !4F
345     @points                        !D0
346   ENDIF                            !60
347   IF br(x-1,y)=2                   !E5
348     br(x-1,y)=5                    !4F
349     sc=sc+500                      !3E
350     stop=35                        !32
351     @points                        !D0
352   ENDIF                            !60
353   @affsp(pac,490000,8,14)          !FC
354   x=x-1                            !7E
355   SUB pac,8                        !73
356   gauche1:                         !B9
357 RETURN                             !9E
358 PROCEDURE haut                     !C7
359   IF br(x,y-1)=1                   !47
360     GOTO haut1                     !B5
361   ENDIF                            !60
362   IF br(x,y-1)=0                   !37
363     br(x,y-1)=5                    !C1
364     @points                        !D0
365   ENDIF                            !60
366   IF br(x,y-1)=2                   !57
367     br(x,y-1)=5                    !C1
368     sc=sc+500                      !3E
369     stop=35                        !32
370     @points                        !D0
371   ENDIF                            !60
372   @affsp(pac,490000,8,14)          !FC
373   y=y-1                            !86
374   SUB pac,2400                     !B4
375   haut1:                           !4A
376 RETURN                             !9E
377 PROCEDURE bas                      !12
378   IF br(x,y+1)=1                   !2F
379     GOTO bas1                      !4B
380   ENDIF                            !60
381   IF br(x,y+1)=0                   !1F
382     br(x,y+1)=5                    !AB
383     @points                        !D0
384   ENDIF                            !60
385   IF br(x,y+1)=2                   !3F
386     br(x,y+1)=5                    !AB
387     sc=sc+500                      !3E
388     stop=35                        !32
389     @points                        !D0
```

```
390    ENDIF                          !60
391    @affsp(pac,490000,8,14)        !FC
392    y=y+1                          !7A
393    ADD pac,2400                   !44
394    bas1:                          !0A
395 RETURN                            !9E
396 PROCEDURE testbord                !AD
397    IF br(x,y)=0                   !70
398      br(x,y)=5                    !4E
399      @points                      !D0
400    ENDIF                          !60
401 RETURN                            !9E
402 PROCEDURE points                  !D5
403    @son(s1$)                      !48
404    sc=sc+176                      !6F
405    @score                         !C1
406    past=past-1                    !DD
```




```
407    IF past=0                      !60
408      ch=1                         !D7
409    ENDIF                          !60
410 RETURN                            !9E
411 PROCEDURE son(a$)                 !10
412    WAVE 0,0                       !89
413    VOID XBIOS(32,L:VARPTR(a$))    !6E
414 RETURN                            !9E
415 PROCEDURE vie                     !98
416    TEXT 172,170,"LIVES"           !7D
417    TEXT 220,170,vie               !E4
418 RETURN                            !9E
419 PROCEDURE score                   !DA
420    TEXT 40,170,"SCORE "           !D8
421    TEXT 88,170,sc                 !F2
422 RETURN                            !9E
423 PROCEDURE affel(ae,as,la,ha)      !FC
424    re=br(xx(num),yy(num))         !AA
425    IF re=0                        !01
426      as=480224                    !26
427    ENDIF                          !60
428    IF re=2                        !12
429      as=480560                    !43
430    ENDIF                          !60
431    @affsp(ae,as,la,ha)            !14
432 RETURN                            !9E
433 PROCEDURE affsp(ae,as,la,ha)      !FC
434    la=(la/4)-1                    !5B
435    LPOKE 150030,ae                !6E
436    LPOKE 150036,as                !B0
437    POKE 150043,la                 !B7
438    POKE 150047,ha-1               !20
439    affiche=150000                 !F8
440    CALL affiche                   !EC
441 RETURN                            !9E
442 PROCEDURE scroll                  !01
443    FOR t=1 TO 25                  !A8
444      PRINT CHR$(10)               !4B
445    NEXT t                         !93
446 RETURN                            !9E
447 PROCEDURE mess(cx,cy,c1,c2,a$)    !01
448    GRAPHMODE 1                    !70
449    DEFTEXT c1,0,0,13              !A2
450    TEXT cx,cy,a$                  !A8
451    GRAPHMODE 2                    !7D
452    DEFTEXT c2,0,0,13              !AE
453    TEXT cx+1,cy+1,a$              !DE
454 RETURN                            !9E
455 PROCEDURE tableaux                !19
456    past=0                         !5C
457    stop=0                         !85
458    INC ntab                       !69
459    adecr=XBIOS(2)+1608            !86
460    FOR y=1 TO 10                  !72
461      esp=0                        !99
462      FOR x=1 TO 18                !37
463        READ b                     !F7
464        IF b=1                     !B6
465          adsp=480000              !6A
466          nb=1                     !25
467        ENDIF                      !3A
468        IF b=0                     !AA
469          adsp=480224              !FC
470          nb=0                     !19
471          INC past                 !85
472        ENDIF                      !3A
473        IF b=2                     !C2
474          adsp=480336              !45
475          nb=1                     !25
476        ENDIF                      !3A
477        IF b=3                     !CE
478          adsp=480448              !8E
479          nb=1                     !25
480        ENDIF                      !3A
481        IF b=4                     !DA
482          adsp=480560              !2B
483          nb=2                     !31
484          INC past                 !85
485        ENDIF                      !3A
486        IF b=8                     !0A
487          adsp=480112              !B3
488          nb=5                     !55
489        ENDIF                      !3A
490        IF b=9                     !16
491          adsp=490000              !79
492          nb=5                     !55
493        ENDIF                      !3A
494        @affsp(adecr+esp,adsp,8,14
     )                                !DC
495      esp=esp+8                    !65
496      SOUND 1,14,#esp+(y*50),1     !61
497      SOUND 0,0,0,0                !21
```

```
498       br(x,y)=nb                    !8F
499     NEXT x                          !43
500     ADD adecr,2400                  !44
501   NEXT y                            !BB
502   u=XBIOS(2)+25608                  !91
503   @affsp(u,480448,8,14)             !66
504   @affsp(u+136,480336,8,14)         !E1
505   @affsp(u+112,480448,8,14)         !E0
506   @affsp(u+64,480336,8,14)          !B3
507   ADD u,2400                        !FE
508   FOR t=1 TO 18                     !C7
509     @affsp(u,480000,8,14)           !09
510     ADD u,8                         !3E
511   NEXT t                            !93
512   TEXT 264,170,ntab                 !6A
513   @score                            !C1
514   @vie                              !67
515   xx(1)=17                          !84
516   yy(1)=8                           !A3
517   fan(1)=XBIOS(2)+18536             !13
518   xx(2)=2                           !6C
519   yy(2)=8                           !A9
520   fan(2)=XBIOS(2)+18416             !D9
521 RETURN                              !9E
522 son1:                               !85
523 DATA 0,25,1,1,7,254,8,16,11,0,12
    ,16,13,9,130,10,255,0               !31
524 son2:                               !89
525 DATA 0,255,20,80,7,1,8,16,11,0,1
    2,16,13,9,12,80,255,0               !36
526 coul:                               !76
527 DATA &000,&566,&511,&004,&002,&0
    06,&117,&233                        !BF
528 DATA &122,&551,&622,&344,&773,&7
    77,&127,&200                        !12
529 pres:                               !75
530 DATA 2,1,1,1,0,1,1,3,0,1,1,1,0,2
    ,1,1                                !08
531 DATA 0,1,0,1,0,1,0,1,0,1,0,3,0,1
    ,0,1                                !94
532 DATA 0,1,0,1,0,1,0,1,0,1,0,0,0,1
    ,0,3                                !88
533 DATA 2,1,1,3,0,1,1,1,0,1,0,0,0,1
    ,0,1                                !80
534 DATA 0,1,0,0,0,1,0,1,0,1,0,1,0,1
    ,0,1                                !50
535 DATA 0,1,0,0,0,2,0,1,0,2,1,1,0,3
    ,1,1                                !F4
536 tab1:                               !46
537 DATA 2,1,1,1,1,1,1,1,1,1,1,1,1,1
    ,1,1,3                              !54
538 DATA 0,8,0,0,0,0,0,0,0,0,4,0,0,0,0
    ,0,0,0,0,0                          !01
539 DATA 2,1,1,1,1,0,1,1,1,1,1,1,1,0
    ,1,1,1,3                            !24
540 DATA 1,4,0,0,1,0,1,9,9,9,9,9,1,0
    ,1,9,9,1                            !E0
541 DATA 1,0,1,0,1,0,1,1,1,1,1,1,1,0
    ,1,9,9,1                            !0A
542 DATA 1,0,1,0,1,4,0,0,0,0,0,0,0,0
    ,1,1,1,1                            !52
543 DATA 1,0,1,0,1,1,1,1,3,1,1,1,1,0
    ,0,0,4,1                            !22
544 DATA 1,0,1,0,4,0,0,0,0,0,0,0,1,0
    ,0,0,0,1                            !EE
545 DATA 1,0,0,0,0,0,0,0,0,0,4,0,0,0
    ,0,0,0,1                            !F6
546 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1
    ,1,1,1,1                            !FE
547 tab2:                               !4A
548 DATA 2,1,1,1,1,1,1,1,1,1,1,1,1,1
    ,9,1,1,3                            !64
549 DATA 0,8,0,0,0,0,0,4,0,0,0,0,0,0,1
    ,9,1,4,0                            !F6
550 DATA 2,1,1,1,1,1,3,1,1,1,1,1,0,1
    ,9,1,0,3                            !44
551 DATA 9,9,9,9,9,9,9,9,9,9,9,1,0,1
    ,9,1,1,1                            !70
552 DATA 1,1,1,1,1,1,2,1,1,1,1,1,0,1
    ,9,9,9,9                            !92
553 DATA 1,4,0,0,0,0,0,0,0,0,0,0,0,1
    ,9,9,9,9                            !DA
554 DATA 1,0,1,1,1,1,1,1,1,1,1,1,1,1
    ,1,1,1,1                            !F6
555 DATA 1,0,0,0,0,0,0,0,4,1,0,0,0,0
    ,0,0,0,1                            !FE
556 DATA 1,4,0,0,1,0,0,0,0,0,4,1,0,0
    ,0,0,4,1                            !D8
557 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1
    ,1,1,1,1                            !FE
558 tab3:                               !4E
559 DATA 2,1,1,1,1,1,9,9,9,9,9,9,1,1
    ,1,1,1,3                            !A4
560 DATA 0,8,0,4,0,1,9,9,9,9,9,9,1,0
    ,4,0,0,0                            !60
561 DATA 2,1,1,1,0,1,9,1,1,1,1,9,1,0
    ,1,1,1,3                            !96
562 DATA 9,9,9,1,0,1,9,1,4,0,1,9,1,0
    ,1,9,9,9                            !BA
563 DATA 9,9,9,1,0,1,9,1,0,0,1,9,1,0
    ,1,9,9,9                            !62
564 DATA 1,1,1,1,0,1,1,1,0,0,1,1,1,0
    ,1,3,1,1                            !EA
565 DATA 1,4,0,0,0,0,0,0,0,0,0,4,0,0
    ,0,0,4,1                            !B6
566 DATA 1,0,1,1,0,1,1,0,0,0,0,1,1,0
    ,1,1,0,1                            !46
567 DATA 1,0,0,0,0,0,0,4,0,0,0,0,0,0
    ,0,0,0,1                            !D6
568 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1
    ,1,1,1,1                            !FE
569 tab4:                               !52
570 DATA 2,1,1,1,1,1,1,9,9,9,9,1,1,1
    ,1,1,1,3                            !34
571 DATA 0,8,0,4,0,0,1,9,9,9,9,1,4,0
    ,0,0,0,0                            !B2
572 DATA 2,0,0,1,1,0,1,9,9,9,9,1,0,1
    ,1,0,0,3                            !AA
573 DATA 1,0,0,0,1,0,0,1,1,1,1,0,0,1
    ,4,0,0,1                            !A0
574 DATA 1,0,1,0,1,0,0,0,0,0,4,0,0,1
    ,0,1,0,1                            !52
575 DATA 1,0,1,4,1,0,0,0,0,0,0,0,0,1
    ,0,1,0,1                            !1B
576 DATA 1,0,0,0,1,0,0,1,1,1,1,0,0,1
    ,0,0,0,1                            !18
577 DATA 1,0,0,1,1,0,1,9,9,9,9,1,0,1
    ,1,0,0,1                            !54
```

```
578 DATA 3,4,0,0,0,4,1,9,9,9,9,1,0,0 |24
,0,0,4,2
579 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1 |FE
,1,1,1,1
580 tab5: |56
581 DATA 1,1,1,1,1,9,9,9,9,9,9,9,1,1 |B6
,1,1,1,1
582 DATA 1,8,0,4,1,1,3,1,1,1,1,1,1,1 |9F
,4,0,0,1
583 DATA 1,0,0,0,0,0,0,0,4,0,0,0,0,0 |1E
,0,0,0,1
584 DATA 1,4,0,0,1,1,1,1,1,1,1,3,1,1 |FA
,0,0,4,1
585 DATA 1,1,0,1,1,0,0,0,0,0,0,0,4,1 |BA
,1,0,1,1
586 DATA 9,1,0,1,0,0,1,1,0,0,1,1,0,0 |82
,1,0,1,9
587 DATA 1,1,0,1,4,0,1,4,0,0,4,1,0,0 |DB
,1,0,1,1
588 DATA 1,0,0,1,1,1,1,0,2,3,0,1,1,1 |EA
,1,0,4,1
589 DATA 1,4,0,0,0,0,0,0,0,0,0,0,0,0 |AE
,0,0,0,1
590 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1 |FE
,1,1,1,1
591 tab6: |66
592 DATA 1,1,1,1,1,3,1,1,1,1,9,9,9,9 |BE
,9,1,1,1
593 DATA 1,8,0,0,0,0,0,0,4,1,9,9,9,9 |EC
,9,1,0,1
594 DATA 1,0,1,0,0,0,0,0,0,0,1,9,9,9 |E4
,9,1,0,1
595 DATA 1,0,3,0,1,0,0,0,4,1,9,9,9,9 |27
,9,1,0,1
596 DATA 1,0,1,0,3,0,1,0,0,1,9,9,9,9 |40
,9,1,0,1
597 DATA 1,0,3,0,1,0,3,0,0,1,9,9,9,9 |BC
,9,1,0,1
598 DATA 1,0,1,0,3,0,1,0,0,1,9,9,9,9 |C6
,9,1,0,1
599 DATA 1,0,1,0,1,0,3,0,4,1,1,3,1,3 |D3
,1,1,4,1
600 DATA 1,0,0,4,0,0,0,0,0,0,0,0,0,4 |2E
,0,0,0,1
601 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1 |FE
,1,1,1,1
602 tab7: |71
603 DATA 1,1,1,1,1,3,1,1,1,1,3,1,1,1 |7C
,1,1,1,1
604 DATA 1,8,0,0,0,0,0,0,0,0,0,0,0,0 |B0
,0,0,0,1
605 DATA 1,0,1,1,1,3,1,1,1,0,3,1,1,1 |FA
,1,1,0,1
606 DATA 1,0,1,9,9,9,9,9,1,4,1,9,9,9 |EA
,9,1,0,1
607 DATA 1,0,1,9,9,9,9,9,1,0,1,9,9,9 |74
,9,1,0,1
608 DATA 1,4,1,1,1,1,1,1,1,0,1,1,1,1 |5A
,1,1,4,1
609 DATA 1,0,0,0,0,0,0,0,0,0,0,0,0,0 |50
,0,0,0,1
610 DATA 1,0,1,1,1,1,1,1,1,1,0,1,1,1 |8E
,1,1,0,1
611 DATA 1,0,0,0,0,0,0,0,0,0,0,0,0,0 |B8
,0,0,0,1
612 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1 |8E
,1,1,1,1
613 tab8: |62
614 DATA 1,1,1,1,1,3,3,1,1,1,1,3,3,1 |B6
,1,1,1,1
615 DATA 1,8,0,0,4,0,0,0,0,0,0,0,0,0 |9F
,0,0,4,1
616 DATA 1,0,1,1,1,1,0,0,1,1,1,3,3,1 |1E
,1,1,0,1
617 DATA 1,0,1,9,9,9,3,0,0,3,9,9,9,9 |FA
,9,1,0,1
618 DATA 1,0,1,9,9,9,9,1,4,0,1,9,9,9 |BA
,9,1,0,1
619 DATA 1,0,1,9,9,9,9,9,3,0,0,3,9,9 |82
,9,1,0,1
620 DATA 1,0,1,9,9,9,9,9,9,1,0,0,1,9 |DB
,9,1,0,1
621 DATA 1,0,1,1,1,1,1,1,1,1,3,0,0,2 |EA
,1,1,0,1
622 DATA 1,4,0,0,0,0,0,0,0,0,0,0,0,0 |AE
,0,0,0,1
623 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1 |FE
,1,1,1,1
624 tab9: |66
625 DATA 1,1,9,9,9,9,1,1,1,1,1,9,9 |BE
,9,9,1,1
626 DATA 1,8,1,9,9,9,1,4,0,0,0,1,9,9 |EC
,9,1,4,1
627 DATA 1,4,0,1,9,9,9,1,0,0,1,9,9,9 |E4
,1,0,0,1
628 DATA 1,0,0,0,1,9,9,1,0,0,1,9,9,1 |27
,0,0,0,1
629 DATA 1,0,0,0,4,1,9,1,0,0,1,9,1,4 |40
,0,0,0,1
630 DATA 1,0,0,0,0,0,1,0,0,0,0,1,0,0 |BC
,0,0,0,1
631 DATA 1,0,0,0,4,0,0,0,0,0,0,0,0,0 |C6
,0,0,0,1
632 DATA 1,0,0,1,2,3,1,0,0,0,0,1,2,3 |D3
,1,0,0,1
633 DATA 1,4,0,1,9,9,1,4,0,0,0,1,9,9 |2E
,1,0,4,1
634 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1 |FE
,1,1,1,1
635 tab10: |71
636 DATA 2,1,1,1,1,1,1,3,1,1,1,1,1,1 |7C
,1,1,1,3
637 DATA 0,0,0,0,0,0,0,4,0,0,0,0,0,0 |B0
,0,0,0,0
638 DATA 2,0,1,1,1,1,0,1,1,1,0,0,1,1 |FA
,1,4,0,1
639 DATA 1,0,1,4,0,0,0,1,0,4,1,0,1,9 |EA
,9,1,0,1
640 DATA 1,0,1,1,1,0,0,1,0,0,1,0,1,9 |74
,9,1,0,1
641 DATA 1,0,1,0,0,0,0,1,0,0,1,0,1,9 |5A
,9,1,0,1
642 DATA 1,0,1,1,1,1,0,1,0,0,1,0,1,1 |50
,1,0,0,1
643 DATA 1,0,0,0,0,0,0,0,0,0,0,0,0,0 |8E
,0,0,0,1
644 DATA 1,4,4,4,4,4,4,4,4,4,4,4,4,4 |4E
,4,4,4,1
```

```
645 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1
    ,1,1,1,1                      !FE
646 routine:                      !BA
647 DATA 601A,0000,0036,0000,0000,00
    00,0000                       !4C
648 DATA 0000,0000,0000,0000,0000,00
    00,FFFF                       !C3
649 DATA 207C,0007,8000,227C,0003,0D
    40,303C                       !8D
650 DATA 000F,323C,001E,48A7,8000,48
    E7,0080                       !02
651 DATA 20D9,51C8,FFFC,4CDF,0100,4C
    9F,0001                       !56
652 DATA D1FC,0000,00A0,51C9,FFE2,4E
    75,0000                       !B8
653 dessins:                      !F0
654 DATA 0000,0000,0000,7FFC,7FF8,43
    F8,21F8                       !1B
655 DATA E206,7FFC,3FFC,401C,FFE2,7F
    FC,3FFC                       !B8
656 DATA 000C,BFF2,7FFC,3FFC,0004,BF
    FA,7FFC                       !20
657 DATA 3FFC,0004,BFFA,7FFC,7FFC,00
    04,FFFA                       !FC
658 DATA 7FFC,7FFC,0004,FFFA,7FFC,7F
    FC,4000                       !28
659 DATA BFFE,7FFC,7FF8,4000,BFFA,7F
    FC,7FF8                       !43
660 DATA 6000,9FFA,7FFC,7FF8,6004,9F
    FE,3FFC                       !D0
661 DATA 3FC4,3F08,C0CE,0000,0000,00
    00,7FFC                       !48
662 DATA 4004,4004,4004,4004,F7CE,B7
    CA,F7CE                       !10
663 DATA F7CE,6FF4,4824,582C,7C7C,3F
    F8,2008                       !F5
664 DATA 2008,3AB8,6FFC,4284,5444,70
    1C,7C7C                       !D9
665 DATA 4444,4BA4,711C,77DC,4004,48
    24,7FFC                       !A2
666 DATA 739C,4004,4004,739C,701C,40
    04,4824                       !0A
667 DATA 783C,5834,600C,4444,7C7C,24
    48,3838                       !5D
668 DATA 2388,27C8,1830,1FF0,1830,18
    30,07C0                       !B7
669 DATA 07C0,07C0,07C0,0000,0000,00
    00,0000                       !C0
670 DATA 0000,0000,0000,0000,0000,00
    00,0000                       !DF
671 DATA 0000,0000,0000,0000,0000,07
    80,0780                       !FA
672 DATA 0780,0780,0840,0FC0,0840,0B
    40,1320                       !60
673 DATA 1CE0,1120,15A0,1420,1BE0,14
    20,1FE0                       !12
674 DATA 1020,1FE0,1020,1760,1020,1F
    E0,1020                       !DC
675 DATA 1220,0840,0FC0,0840,0840,07
    80,0780                       !D2
676 DATA 0780,0780,0000,0000,0000,00
    00,0000                       !0C
677 DATA 0000,0000,0000,0000,0000,00
    00,0000                       !DF
678 DATA 0000,0000,0000,7FFC,7FF8,43
    F8,21F8                       !1B
679 DATA E206,7C7C,3C7C,401C,FC62,78
    7C,387C                       !AD
680 DATA 000C,B872,700C,320C,0204,B0
    0A,600C                       !0A
681 DATA 260C,07E4,A20A,400C,4FCC,0F
    E4,CFCA                       !23
682 DATA 400C,4FCC,0FE4,CC0A,600C,66
    0C,47E0                       !50
683 DATA A00E,700C,7208,4200,B00A,78
    7C,7878                       !70
684 DATA 6000,987A,7C7C,7C78,6004,9C
    7E,3FFC                       !3A
685 DATA 3FC4,3F08,C0CE,0000,0000,00
    00,7FFC                       !48
686 DATA 0000,0000,0000,7FFC,7FF8,43
    F8,21F8                       !1B
687 DATA E206,7C7C,3C7C,401C,FC62,7C
    3C,3C3C                       !88
688 DATA 018C,BC32,601C,209C,01C4,A0
    1A,600C                       !2D
689 DATA 20CC,0FE4,A04A,6004,67E4,0F
    F0,E7E6                       !A7
690 DATA 6004,67E4,0FF4,E042,600C,60
    CC,4FE0                       !EB
691 DATA A00E,601C,6098,41C0,A01A,7C
    3C,7C38                       !5E
692 DATA 6180,9C3A,7C7C,7C78,6004,9C
    7E,3FFC                       !D1
693 DATA 3FC4,3F08,C0CE,0000,0000,00
    00,7FFC                       !48
694 DATA 0000,0000,0000,0000,0000,00
    00,0000                       !DF
695 DATA 0000,0000,0000,0000,0000,07
    80,0780                       !FA
696 DATA 0000,0000,0840,0FC0,0780,03
    00,1320                       !24
697 DATA 1CE0,0DC0,0580,1420,1BE0,0F
    C0,0FC0                       !F8
698 DATA 1020,1FE0,0FC0,0740,1020,1F
    E0,0FC0                       !C1
699 DATA 0200,0840,0FC0,0780,0000,07
    80,0780                       !BA
700 DATA 0000,0000,0000,0000,0000,00
    00,0000                       !DF
701 DATA 0000,0000,0000,0000,0000,00
    00,0000                       !DF
702 DATA 0000,0000,0000,07C0,07C0,05
    40,0000                       !25
703 DATA 0D60,0FE0,0820,0000,1830,1B
    A0,12A0                       !2A
704 DATA 0000,12B8,1920,1020,0000,30
    38,3930                       !67
705 DATA 36D0,0000,345C,7FF8,4288,00
    00,428C                       !78
706 DATA FC7C,8444,0000,8444,F83C,80
    00,0000                       !02
707 DATA 8000,FBBC,D004,8000,5004,7F
    FC,780C                       !F5
708 DATA 7004,8808,3FFC,3FFC,399C,86
    60,4672                       !7D
709 DATA 4672,4672,318C,0000,0000,00
    00,4E72                       !81
```

Tridimensionnellement vôtre

VOLUMIX (GFA BASIC)

Ce logiciel permet l'élaboration de figures en fil de fer dans un espace tridimensionnel et leur rotation selon les axes x,y,z.

Il se compose de 2 programmes: VOLUMIX qui réserve de la mémoire puis charge VOLUMIX1, le programme principal.

Menu fichier

Load: charge un fichier précédemment créé.
Save: sauvegarde une figure.
Save cycle: sauvegarde une cycle complet si vous avez précédemment sélectionné Start. Les fichiers cycles créés par VOLUMIX peuvent être utilisés dans vos propres programmes.
Quit: quitte VOLUMIX.

Menu création

Point: place un point aux coordonnées X,Y,Y du viseur actuel.
Connection: si vous avez plus de 2 points, déplacez un petit viseur avec le bouton droit de la souris. Une fois sur le bon point, cliquez sur le bouton gauche. Apparaît alors une ligne du point actuel jusqu'au point suivant. Sélectionnez le second point de la même manière. Quitter cette fonction par une touche quelconque du clavier.
Détruire point, détruire connection: affiche un point entouré ou une ligne en vert pour la sélection et la destruction. Même méthode d'utilisation de la souris que dans connection.
Sym H: chaque point que vous tracez est dédoublé horizontalement.
Sym V: même chose mais verticalement. Si vous sélectionner Sym V et Sym H, vous obtiendrez 3 points lorsque vous en tracerez un.
Clear: mise à zéro de tous les pointeurs.

Menu vue

Vue de face: montre la figure actuelle en vue de face.
Vue de dessus: idem en vue de dessus.
Vue de gauche: idem en vue de gauche.

Menu rotation

Angle X: rotation selon l'axe horizontal.
Angle Y: rotation selon l'axe vertical.
Angle Z: idem selon l'axe de la profondeur.
Start: effectue les calculs et montre la figure actuelle en mouvement.

Daniel Provenier

```
' ▪ indique l'endroit où
' vous devez frapper Return.
'
RESERVE 100000▪
RUN "volumix1"▪



' ▪ indique l'endroit où
' vous devez frapper Return.
'
DEFWRD "a-z"▪
@init▪
@prog▪
EDIT▪
PROCEDURE prog▪
 ON MENU GOSUB qqc▪
 WHILE qut&=0▪
   SLEEP▪
   IF MOUSEK=1▪
     @viseur▪
     @viseur1▪
   ENDIF▪
   @coord▪
 WEND▪
RETURN▪
PROCEDURE qqc▪
 m&=MENU(0)▪
 ON m&+1 GOSUB 0,load,save,savec,quit,0,
0,point,connec,destp,destc,symh,symv,cle
ar,0,0,vuef,vued,vueg,0,0,angx,angy,angz
,start,0▪
RETURN▪
PROCEDURE init▪
 OPENS 1,0,0,320,250,2,0▪
 OPENW #1,0,0,320,250,0,0▪
 CLEARW #1▪
 TITLEW #1,"ROTATION 3D"▪
 adr1%=WINDOW(1)+50▪
 RASTPORT LONG(adr1%)▪
 SETCOLOR 0,0,0,0▪
 SETCOLOR 1,15,15,15▪
 SETCOLOR 2,0,15,0▪
 SETCOLOR 3,0,0,15▪
 DEFINT "a-z"▪
 DEFFLT "c,s"▪
 DIM x&(200),y&(200),z&(200),x1&(70,200)
,y1&(70,200),j&(1,200),a$(25),valm&(25)▪
 np&=0▪
 nc&=0▪
 vue&=1▪
 quit&=0▪
 x&=160▪
 y&=100▪
 z&=160▪
 ax&=5▪
 ay&=5▪
 az&=5▪
 rx&=0▪
 ry&=0▪
 rz&=0▪
 symh%=0▪
```

```
symv%=0▪
FOR n&=0 TO 25▪
  READ a$(n&)▪
  valm&(n&)=&H52▪
NEXT n&▪
MENU a$()▪
FOR n&=21 TO 23▪
  valm&(n&)=valm&(n&) OR &H101▪
  MENU n&,valm&(n&)▪
NEXT n&▪
valm&(16)=valm&(16) OR &H101▪
MENU 16,valm&(16)▪
@trait_axe▪
@aff_face▪
@viseur1▪
DATA FICHIER,Load,Save,Save Cycle,Quit,
"",▪
DATA CREATION,Point,Connection,Detruire
point,Detruire connec.,  Sym H,  Sym V,
Clear,"",▪
DATA VUE,  Face,  Dessus,  Gauche,"",▪
DATA ROTATION,  Angle x,  Angle y,  Ang
le z,  Start,"",▪
DATA "",▪
RETURN▪
PROCEDURE calcul▪
@menu_off▪
cy&=1▪
CLS▪
BOX 0,0,319,200▪
LOCATE 1,1▪
PRINT "Please Wait"▪
PRINT "Computing    positions"▪
rx&=0▪
ry&=0▪
rz&=0▪
FOR i&=0 TO 70▪
  LOCATE 11,2▪
  PRINT USING "##",71-i&;▪
  cx=COSQ(rx&)▪
  ssx=SINQ(rx&)▪
  cy=COSQ(ry&)▪
  ssy=SINQ(ry&)▪
  cz=COSQ(rz&)▪
  ssz=SINQ(rz&)▪
  c1=cy*cz▪
  c2=cy*ssz▪
  c3=-ssy▪
  ssy1=-ssy*ssx▪
  c4=ssy1*cz+cx*ssz▪
  c5=ssy1*ssz-cx*cz▪
  c6=-ssx*cy▪
  c7=cx*ssy*cz+ssx*ssz▪
  c8=cx*ssy*ssz-ssx*cz▪
  c9=cx*cy▪
  FOR n&=0 TO np&-1▪
    a1&=x&(n&)▪
    a2&=y&(n&)▪
    a3&=z&(n&)▪
    szz=(a1&*c7+a2&*c8+a3&*c9)+160▪
    x1&(i&,n&)=(a1&*c1+a2&*c2+a3&*c3)*1
60/szz+160▪
    y1&(i&,n&)=-(a1&*c4+a2&*c5+a3&*c6)*
125/szz+125▪
  NEXT n&▪
  rx&=(rx&+ax&)▪
  ry&=(ry&+ay&)▪
  rz&=(rz&+az&)▪
NEXT i&▪
@menu_on▪
RETURN▪
PROCEDURE affiche▪
OPENS 2,0,0,320,250,1,0▪
OPENW #2,0,0,320,200,0,0,2▪
OPENW #3,0,0,320,200,0,0,2▪
CLEARW #2▪
CLEARW #3▪
FRONTW #2▪
adr3%=(WINDOW(3)+50)▪
adr2%=(WINDOW(2)+50)▪
WHILE MOUSEK<>1▪
  i&=-1▪
  WHILE MOUSEK<>1 AND i&<70▪
    INC i&▪
    RASTPORT adr3%▪
    CLEARW #3▪
    FOR n&=0 TO nc&-1▪
      LINE x1&(i&,j&(0,n&)),y1&(i&,j&(0
,n&)),x1&(i&,j&(1,n&)),y1&(i&,j&(1,n&))▪
    NEXT n&▪
    GET 0,0,320,200,i$▪
    RASTPORT adr2%▪
    PUT 0,0,i$▪
  WEND▪
WEND▪
CLOSES 2▪
RASTPORT LONG(WINDOW(1)+50)▪
@trait_axe▪
@aff_face▪
ras=1▪
RETURN▪
PROCEDURE trait_axe▪
CLS▪
COLOR 1▪
BOX 0,0,319,200▪
COLOR 2▪
FOR i&=-15 TO 199 STEP 20▪
  LINE 160,i&,160,i&+10▪
  PLOT 160,i&+15▪
NEXT i&▪
FOR i&=5 TO 319 STEP 20▪
  LINE i&,100,i&+10,100▪
  PLOT i&+15,100▪
NEXT i&▪
COLOR 1▪
RETURN▪
PROCEDURE aff_face▪
IF vue&=1▪
  FOR n&=0 TO np&-1▪
    PLOT x&(n&)+160,y&(n&)+100▪
  NEXT n&▪
  FOR n&=0 TO nc&-1▪
    LINE x&(j&(0,n&))+160,y&(j&(0,n&))+
100,x&(j&(1,n&))+160,y&(j&(1,n&))+100▪
  NEXT n&▪
ELSE IF vue&=3▪
  FOR n&=0 TO np&-1▪
    PLOT 160-z&(n&),y&(n&)+100▪
  NEXT n&▪
  FOR n&=0 TO nc&-1▪
    LINE 160-z&(j&(0,n&)),y&(j&(0,n&))+
100,160-z&(j&(1,n&)),y&(j&(1,n&))+100▪
```

```
    NEXT n&
  ELSE
    FOR n&=Ø TO np&-1
      PLOT x&(n&)+16Ø,1ØØ-z&(n&)
    NEXT n&
    FOR n&=Ø TO nc&-1
      LINE x&(j&(Ø,n&))+16Ø,1ØØ-z&(j&(Ø,n
&)),x&(j&(1,n&))+16Ø,1ØØ-z&(j&(1,n&))
    NEXT n&
  ENDIF
RETURN
PROCEDURE start
  IF nc&<>Ø
    @calcul
    @affiche
    @viseur
  ELSE
    PRINT CHR$(7)
  ENDIF
RETURN
PROCEDURE quit
  qut&=1
RETURN
PROCEDURE angx
  ax&=ax& XOR 5
  n&=ax&
  @menuang
RETURN
PROCEDURE angy
  ay&=ay& XOR 5
  n&=ay&
  @menuang
RETURN
PROCEDURE angz
  az&=az& XOR 5
  n&=az&
  @menuang
RETURN
PROCEDURE menuang
  IF n&=Ø
    valm&(m&)=valm&(m&) AND &HFE
  ELSE
    valm&(m&)=valm&(m&) OR &H1Ø1
  ENDIF
  MENU m&,valm&(m&)
RETURN
PROCEDURE viseur
  GRAPHMODE 3
  COLOR 3
  LINE x1&-5,y1&,x1&+5,y1&
  LINE x1&,y1&-5,x1&,y1&+5
  COLOR 1
  GRAPHMODE Ø
RETURN
PROCEDURE viseur1
  GRAPHMODE 3
  COLOR 3
  x1&=MOUSEX
  y1&=MOUSEY
  rep&=y1&
  LINE x1&-5,y1&,x1&+5,y1&
  LINE x1&,y1&-5,x1&,y1&+5
  COLOR 1
  GRAPHMODE Ø
  IF vue&=1
    x&=x1&
    y&=y1&
  ELSE IF vue&=2
    x&=x1&
    z&=y1&
  ELSE
    y&=y1&
    z&=x1&
  ENDIF
RETURN
PROCEDURE viseur2
  IF vue&=1
    x1&=x&
    y1&=y&
  ELSE IF vue&=2
    x1&=x&
    z&=z&-6Ø
    y1&=z&
  ELSE
    x1&=z&
    y1&=y&
  ENDIF
RETURN
PROCEDURE vuef
  IF vue&<>1
    valm&(m&)=valm&(m&) OR &H1Ø1
    valm&(m&+1)=valm&(m&+1) AND &HFE
    valm&(m&+2)=valm&(m&+2) AND &HFE
    FOR n&=m& TO m&+2
      MENU n&,valm&(n&)
    NEXT n&
    IF vue&=2
      z&=z&+6Ø
    ENDIF
    vue&=1
    @trait_axe
    @aff_face
    @viseur2
    @viseur
  ENDIF
RETURN
PROCEDURE vued
  IF vue&<>2
    valm&(m&)=valm&(m&) OR &H1Ø1
    valm&(m&-1)=valm&(m&-1) AND &HFE
    valm&(m&+1)=valm&(m&+1) AND &HFE
    FOR n&=m&-1 TO m&+1
      MENU n&,valm&(n&)
    NEXT n&
    vue&=2
    @trait_axe
    @aff_face
    @viseur2
    @viseur
  ENDIF
RETURN
PROCEDURE vueg
  IF vue&<>3
    valm&(m&)=valm&(m&) OR &H1Ø1
    valm&(m&-2)=valm&(m&-2) AND &HFE
    valm&(m&-1)=valm&(m&-1) AND &HFE
    FOR n&=m&-2 TO m&
      MENU n&,valm&(n&)
    NEXT n&
    IF vue&=2
      z&=z&+6Ø
    ENDIF
```

```
    vue&=3*
    @trait_axe*
    @aff_face*
    @viseur2*
    @viseur*
  ENDIF*
RETURN*
PROCEDURE point*
 IF rep&<200*
   PLOT x1&,y1&*
   @point1*
   IF symh%=1*
     PLOT 320-x1&,y1&*
     x&=320-x&*
     @point1*
     x&=320-x&*
   ENDIF*
   IF symv%=1*
     PLOT x1&,200-y1&*
     y&=200-y&*
     @point1*
     y&=200-y&*
   ENDIF*
 ELSE*
   PRINT CHR$(7);*
 ENDIF*
RETURN*
PROCEDURE point1*
 ON vue& GOSUB pointf,pointd,pointg*
 x&(np&)=x2&*
 y&(np&)=y2&*
 z&(np&)=z2&*
 np&=np&+1*
RETURN*
PROCEDURE pointf*
 x2&=x&-160*
 y2&=y&-100*
 z2&=160-z&*
RETURN*
PROCEDURE pointd*
 x2&=x&-160*
 y2&=y&-100*
 z2&=100-z&*
RETURN*
PROCEDURE pointg*
 x2&=x&-160*
 y2&=y&-100*
 z2&=160-z&*
RETURN*
PROCEDURE connec*
 @menu_off*
 IF np&>1*
   GRAPHMODE 3*
   test&=0*
   xx&=0*
   @box*
   GRAPHMODE 3*
   WHILE INKEY$<>CHR$(27)*
     IF test&=1*
       @line*
       @line*
     ENDIF*
     a&=MOUSEK*
     IF a&=2*
       @box*
       INC xx&*
       IF xx&=np&*
         xx&=0*
       ENDIF*
       @box*
       @pause*
     ELSE IF a&=1*
       @box*
       IF test&=0*
         test&=1*
         xx1&=xx&*
         @box*
         @pause*
       ELSE*
         test&=0*
         GRAPHMODE 0*
         @line*
         j&(0,nc&)=xx&*
         j&(1,nc&)=xx1&*
         INC nc&*
         GRAPHMODE 3*
         @box*
         @pause*
       ENDIF*
     ENDIF*
   WEND*
   @box*
   GRAPHMODE 0*
 ENDIF*
 menu_on*
RETURN*
PROCEDURE box*
 IF vue&=1*
   BOX x&(xx&)+158,y&(xx&)+98,x&(xx&)+162,y&(xx&)+102*
 ENDIF*
 IF vue&=3*
   BOX 162-z&(xx&),y&(xx&)+98,158-z&(xx&),y&(xx&)+102*
 ENDIF*
 IF vue&=2*
   BOX x&(xx&)+158,102-z&(xx&),x&(xx&)+162,98-z&(xx&)*
 ENDIF*
RETURN*
PROCEDURE line*
 IF vue&=1*
   LINE x&(xx&)+160,y&(xx&)+100,x&(xx1&)+160,y&(xx1&)+100*
 ENDIF*
 IF vue&=3*
   LINE 160-z&(xx&),y&(xx&)+100,160-z&(xx1&),y&(xx1&)+100*
 ENDIF*
 IF vue&=2*
   LINE x&(xx&)+160,100-z&(xx&),x&(xx1&)+160,100-z&(xx1&)*
 ENDIF*
RETURN*
PROCEDURE pause*
 WHILE MOUSEK<>0*
 WEND*
RETURN*
PROCEDURE clear*
 ALERT 1,"Clear|are your|sure !",1,"Yes|No",a&*
 IF a&=1*
```

```
  np&=0
  nc&=0
  @trait_axe
  @viseur
  cy&=0
 ENDIF
 FRONTS 1
RETURN
PROCEDURE load
 @menu_off
 IF ras=1
   BACKS 1
 ENDIF
 FILESELECT "Fichier 3D","Charger","DF0:
",nom$
 IF nom$<>""
   OPEN "I",#1,nom$
   INPUT #1,a$
   IF a$="VOLUMIX"
     INPUT #1,nc&
     INPUT #1,np&
     FOR n&=0 TO np&-1
       INPUT #1,x&(n&),y&(n&),z&(n&)
     NEXT n&
     FOR n&=0 TO nc&-1
       INPUT #1,j&(0,n&),j&(1,n&)
     NEXT n&
     CLOSE #1
     @trait_axe
     @aff_face
     @viseur
   ENDIF
   CLOSE #1
 ENDIF
 FRONTS 1
 @menu_on
RETURN
PROCEDURE save
 @menu_off
 IF ras=1
   BACKS 1
 ENDIF
 FILESELECT "Fichierr 3D","Sauver","",no
m$
 IF nom$<>""
   OPEN "O",#1,nom$
   WRITE #1,"VOLUMIX"
   WRITE #1,nc&
   PRINT #1,np&
   FOR n&=0 TO np&-1
     WRITE #1,x&(n&),y&(n&),z&(n&)
   NEXT n&
   FOR n&=0 TO nc&-1
     WRITE #1,j&(0,n&),j&(1,n&)
   NEXT n&
   CLOSE #1
 ENDIF
 FRONTS 1
 @menu_on
RETURN
PROCEDURE symv
 symv%=symv% XOR 1
 IF symv%=1
   valm&(m&)=valm&(m&) OR &H101
 ELSE
   valm&(m&)=valm&(m&) AND &HFE
 ENDIF
 MENU m&,valm&(m&)
RETURN
PROCEDURE symh
 symh%=symh% XOR 1
 IF symh%=1
   valm&(m&)=valm&(m&) OR &H101
 ELSE
   valm&(m&)=valm&(m&) AND &HFE
 ENDIF
 MENU m&,valm&(m&)
RETURN
PROCEDURE destp
 @menu_off
 IF np&<>0
   GRAPHMODE 3
   xx&=0
   @box
   GRAPHMODE 3
   WHILE INKEY$="" AND np&<>0
     a&=MOUSEK
     IF a&=2
       @box
       INC xx&
       IF xx&=np&
         xx&=0
       ENDIF
       @box
       @pause
     ELSE IF a&=1
       @box
       DEC np&
       DELETE x&(xx&)
       DELETE y&(xx&)
       DELETE z&(xx&)
       '
       FOR n=0 TO nc&-1
         IF j&(0,n)=xx& OR j&(1,n)=xx&
           FOR k=n TO nc&-2
             j&(0,k)=j&(0,k+1)
             j&(1,k)=j&(1,k+1)
           NEXT k
           j&(0,k)=0
           j&(1,k)=0
           DEC nc&
         ENDIF
         IF j&(0,n)>xx&
           DEC j&(0,n)
         ELSE IF j&(1,n)>xx&
           DEC j&(1,n)
         ENDIF
       NEXT n
       GRAPHMODE 0
       @trait_axe
       @aff_face
       @viseur
       GRAPHMODE 3
       @box
       @pause
     ENDIF
   WEND
   @box
   GRAPHMODE 0
 ENDIF
 @menu_on
RETURN
```

```
PROCEDURE destc•
 @menu_off•
 IF np&<>0•
   GRAPHMODE 3•
   xx&=0•
   @line1•
   GRAPHMODE 3•
   WHILE INKEY$="" AND nc&<>0•
     a&=MOUSEK•
     IF a&=2•
       @line1•
       INC xx&•
       IF xx&=nc&•
         xx&=0•
       ENDIF•
       @line1•
       @pause•
     ELSE IF a&=1•
       •
       FOR k=xx& TO nc&-1•
         j&(0,k)=j&(0,k+1)•
         j&(1,k)=j&(1,k+1)•
       NEXT k•
       j&(0,k)=0•
       j&(1,k)=0•
       DEC nc&•
       GRAPHMODE 0•
       @trait_axe•
       @aff_face•
       @viseur•
       GRAPHMODE 3•
       @pause•
       @line1•
     ENDIF•
   WEND•
   @line1•
   GRAPHMODE 0•
 ENDIF•
 @menu_on•
RETURN•
PROCEDURE line1•
 IF vue&=1•
   LINE x&(j&(0,xx&))+160,y&(j&(0,xx&))+
100,x&(j&(1,xx&))+160,y&(j&(1,xx&))+100•
 ENDIF•
 IF vue&=3•
   LINE 160-z&(j&(0,xx&)),y&(j&(0,xx&))+
100,160-z&(j&(1,xx&)),y&(j&(1,xx&))+100•
 ENDIF•
 IF vue&=2•
   LINE x&(j&(0,xx&))+160,100-z&(j&(0,xx
&)),x&(j&(1,xx&))+160,100-z&(j&(1,xx&))•
 ENDIF•
RETURN•
PROCEDURE menu_on•
 FOR n=1 TO 4•
   @menu2•
 NEXT n•
 FOR n=7 TO 13•
   @menu2•
 NEXT n•
 FOR n=16 TO 18•
   @menu2•
 NEXT n•
 FOR n=21 TO 24•
   @menu2•
 NEXT n•
RETURN•
PROCEDURE menu_off•
 FOR n=1 TO 4•
   @menu1•
 NEXT n•
 FOR n=7 TO 13•
   @menu1•
 NEXT n•
 FOR n=16 TO 18•
   @menu1•
 NEXT n•
 FOR n=21 TO 24•
   @menu1•
 NEXT n•
RETURN•
PROCEDURE menu1•
 valm&(n)=valm&(n) AND &HFEF•
 MENU n,valm&(n)•
RETURN•
PROCEDURE menu2•
 valm&(n)=valm&(n) OR &H10•
 MENU n,valm&(n)•
RETURN•
PROCEDURE coord•
 LOCATE 1,26•
 x2&=x&-160•
 y2&=y&-100•
 IF vue&=1•
   z2&=160-z&•
 ELSE IF vue&=2•
   z2&=100-z&•
 ELSE•
   z2&=160-z&•
 ENDIF•
 PRINT USING "x=#### ",x2&•
 PRINT USING "y=#### ",y2&•
 PRINT USING "z=#### ",z2&•
RETURN•
PROCEDURE savec•
 @menu_off•
 IF cy&=1•
   IF ras=1•
     BACKS 1•
   ENDIF•
   FILESELECT "Fichier 3D Cycle","Sauver
","",nom$•
   IF nom$<>""•
     OPEN "O",#1,nom$•
     WRITE #1,"VOLUMIX.CYCLE"•
     WRITE #1,nc&•
     PRINT #1,np&•
     FOR i&=0 TO 70•
       FOR n&=0 TO np&-1•
         WRITE #1,x1&(i&,n&),y1&(i&,n&)•
       NEXT n&•
     NEXT i&•
     FOR n&=0 TO nc&-1•
       WRITE #1,j&(0,n&),j&(1,n&)•
     NEXT n&•
     CLOSE #1•
   ENDIF•
   FRONTS 1•
 ENDIF•
 @menu_on•
RETURN•
```

Un bon coup de poignet

JOYSTICK AUTOMATIQUE

Qui n'a eu, au détour d'un «Summer Games» ou d'un «Combat School», le bras complètement ankylosé après avoir réduit en bouillie les microswitches de son joystick?

Terminé la galère grâce à un petit montage facile à réaliser pour peu que l'on possède un minimum de patience et de matériel.

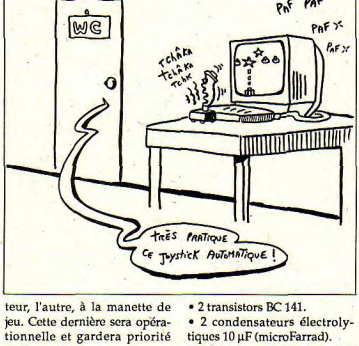
Principe

Lorsqu'un transistor est passant, le condensateur relié à son collecteur se décharge. Le courant issu du + 5 V passe alors par ce condensateur. La tension de base du second transistor devient donc nulle, ce qui le bloque. L'autre condensateur se charge alors en prenant le «+» venant de la broche reliée au collecteur du transistor bloqué. Une fois le premier condensateur complètement déchargé, le courant ne peut plus passer par ses bornes. Il repasse donc par la base du transistor et le second condensateur commence à se décharger. Le cycle recommence.

Mise en pratique

Je vous conseille d'effectuer ce montage sur une plaquette à bandes cuivrées et percées, vendue pour environ 25 F chez tous les revendeurs de composants électroniques. Cette plaquette, une fois tous les composants soudés, est à placer dans une boîte où seront fixés quelques interrupteurs: deux commutateurs à trois positions reliant leurs broches opposées une à une et un interrupteur pour la commande marche/arrêt. Sortiront de cette boîte, deux câbles plats à 6 conducteurs reliés à des fiches 9 broches (une fiche mâle et une femelle). L'une de ces deux fiches est destinée au port joystick de votre ordinateur, l'autre, à la manette de jeu. Cette dernière sera opérationnelle et gardera priorité sur le montage.

Si vous constatez que, après avoir bougé la poignée de votre manette, ou pour une autre raison, le montage reste bloqué dans une position, éteignez-le, puis rallumez-le. Ceci peut être dû à une valeur de charge égale des deux condensateurs.



Liste des composants

- 4 diodes 1N4004.
- 2 résistances 1/4 de watt 10 Kohms (marron, noir, orange).
- 1 résistance 1/4 de watt 5,6 Kohms (vert, bleu, rouge).
- 2 transistors BC 141.
- 2 condensateurs électrolytiques 10 µF (microFarrad).
- 1 plaquette à bandes cuivrées percée.
- 1 interrupteur à fermeture 2 positions.
- 2 commutateurs 3 positions.
- 1 prise 9 broches mâle.
- 1 prise 9 broches femelle.

Test du montage

Une fois monté, il va falloir tester votre montage. Ce court programme en Amiga Basic vous y aidera.

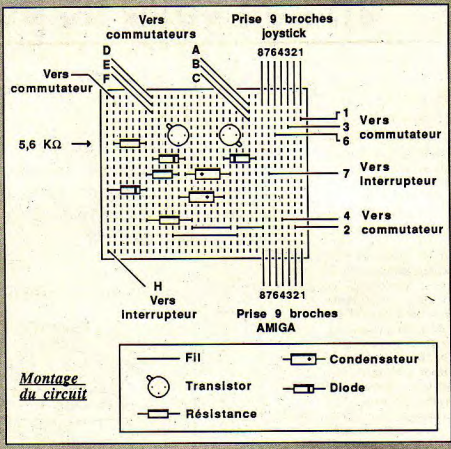
```
CLS
LOCATE 1,20
PRINT "Testeur du montage électronique"
start:
  IF STICK(2)=-1 THEN
    LOCATE 10,5
    PRINT "Gauche"
    WHILE STICK(2)=-1:WEND
    LOCATE 10,5
    PRINT "        "
  END IF
  IF STICK(2)=1 THEN
    LOCATE 10,30
    PRINT "Droite"
    WHILE STICK(2)=1:WEND
    LOCATE 10,30
    PRINT "      "
  END IF
  IF STICK(3)=-1 THEN
    LOCATE 5,15
    PRINT "Haut"
    WHILE STICK(3)=-1:WEND
    LOCATE 5,15
    PRINT "      "
  END IF
  IF STICK(3)=1 THEN
    LOCATE 15,15
    PRINT "Bas"
    WHILE STICK(3)=1:WEND
    LOCATE 15,15
    PRINT "    "
  END IF
  IF STRIG(3)=-1 THEN
    LOCATE 20,2
    PRINT "Tir"
    WHILE STRIG(3)=-1:WEND
    LOCATE 20,2
    PRINT "   "
  END IF
GOTO start
```

Après l'avoir tapé, branchez la prise 9 broches dans le port joystick #2 (celui qui n'est pas occupé par la souris). L'affichage à l'écran doit être le suivant:

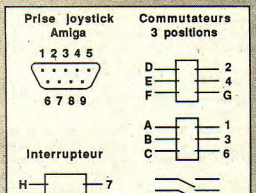
- Un commutateur en position D-2 et l'autre en position A-1: affichage alternatif de «Bas» et «Haut».
- Un commutateur en position E-4 et l'autre en position B-3: affichage alternatif de «Droite» et «Gauche».
- Un commutateur en position F-G et l'autre en position C-6: affichage de «Tir» clignotant rapidement (tir automatique).

Si une valeur s'affiche et reste allumée, éteignez puis rallumez le montage. Si cela ne fonctionne toujours pas, vérifiez soigneusement les branchements. Si rien ne s'affiche, assurez-vous que les broches 7 et 8 du circuit sont bien reliées. Bonne chance!

Didier Arenzana



Montage du circuit



Brochage

Les n° correspondent aux broches, les lettres, aux points du circuit.

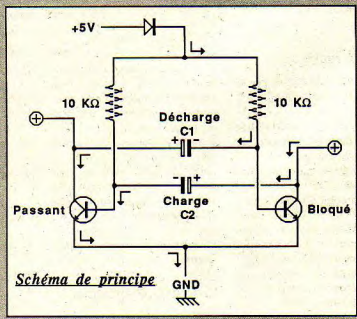


Schéma de principe

Algorithmique

LES TRIS

La plupart des programmes nécessitent tôt ou tard un classement de leurs données par l'obligeance d'un algorithme de tri. Votre langage préféré n'en dispose pas? Programmez-le vous-même...

Le but de cet article est de vous éviter d'avoir à choisir entre le tri à bulles et... le tri à bulles en présentant un certain nombre d'alternatives parmi les plus classiques...

Tout d'abord, pour classer un ensemble d'éléments comme un tableau d'entiers ou une suite de noms, etc., il faut disposer d'une relation d'«ordre» qui permette de décider pour deux éléments quelconques de cet ensemble s'ils sont égaux ou différents et dans ce dernier cas, lequel est le plus grand. On a par exemple la relation «<=» pour les entiers et les réels ou encore l'ordre alphabétique pour les chaînes de caractères. En d'autre termes et pour chaque élément, on doit choisir et disposer d'une information qui le rende comparable à tout autre élément, celle-ci s'appelle la CLE. Ainsi, dans le cas d'un tableau d'entiers, la clé pourra être l'entier lui-même. Pour une fiche statistique comportant nom, adresse, date de naissance..., la clé (le critère) choisie pourra être le nom ou la date de naissance ou le département etc.

Ceci posé, tous les programmes de tri proposés se débrouilleront avec trois paramètres qui seront :

- Le nombre d'éléments à trier.
- Un pointeur sur une routine comparant deux éléments x1 et x2 telle qu'elle renvoie :

- 0, si clé(x1) == clé(x2) ;
- un nombre positif, si clé(x1) > clé(x2) ;
- un nombre négatif, si clé(x1) < clé(x2) ;

Cette routine appelée compare() ou cmp_cles() dans le tri recevra comme paramètre deux entiers permettant d'accéder a x1 et x2 par rapport à une structure de donnée quelconque, tableau ou fichier, etc. Ces entiers seront l'indice dans le cas d'un tableau ou le numéro d'enregistrement pour un fichier en accès direct.

- Un pointeur sur une routine permutant deux élément x1 et x2 et ayant les mêmes paramètres d'entrée que cmp_cle(). Cette routine sera baptisée echange() ou swp_elts() dans le tri.

Les deux routines ci-contre (listing 1) montrent comment peuvent se présenter cmp_cles() et swp_elts() dans un cas non trivial. Ici, il s'agit de classer les éléments d'un objet de type LISTE par ordre alphabétique. La clef est une chaîne contenue dans les variables «info».

Si la méthode ci-dessus paraît lourde et formaliste, elle présente pourtant un avantage énorme: le programme de tri proprement dit est totalement indépendant du type de donnée qu'il doit classer, cette information n'étant utilisée qu'au niveau de cmp_cles() et swp_elts(). Dans ces conditions, on a obtenu un programme de tri universel pouvant aussi bien trier des tableau d'entiers que des tableaux de structures ou des noms ou encore même des fichiers. Les seules choses à modifier en fonction du type de tri souhaité sont les deux fonctions cmp_cle() et swp_elt(). Pour concrétiser les choses, voyons sur un petit exemple *(listing 2)* comment se déroulent les opérations avec un tableau d'entiers qu'on désire trier par ordre croissant.

Note: bien entendu, si vous ne disposez pas d'un langage permettant de passer en paramètre un pointeur sur une fonction, vous pourrez toujours remplacer cmp_cles() ou swp_elts() par leur valeur à l'endroit où ceux-ci apparaissent.

Si tous les algorithmes qui vont suivre sont capables de classer, ils ne sont pas égaux devant le tri ou même selon l'ensemble à trier. Certains deviennent très lents si le nombre d'objets dépasse quelques centaines et carrément rédhibitoires passé quelques milliers (tri à bulles, tri par insertion). D'autres ne sont intéressants que pour des nombres moyens ou grands (tri par arbre ou heapsort). Il en existe même qui, bien que globalement très efficaces (tri rapide ou quicksort), risquent de dégénérer quand les données se présentent dans un certain ordre (même si la probabilité est infime, les lois de Murphy rappellent que quand le pire peut arriver, il

```
cmp_list( i, j )
int i, j ;

{
    LISTE * lst = ( LISTE * ) ptr_tri ;

    return( strcmp( lst->info[i], lst->info[j] ) ) ;
}

swp_list( i, j )
register int i, j ;

{
    register LISTE * lst = ( LISTE * ) ptr_tri ;
    register int attrib ;
    register long ptr ;

    attrib = lst->attrib[ i ] ;
    lst->attrib[i] = lst->attrib[j] ;
    lst->attrib[j] = attrib ;

    ptr = ( long ) lst->info[i] ;
    lst->info[i] = lst->info[j] ;
    lst->info[j] = ( char * ) ptr ;

    ptr = ( long ) lst->pointr[i] ;
    lst->pointr[i] = lst->pointr[j] ;
    lst->pointr[j] = ( long ) ptr ;
}
```

Listing 1

```
typedef int ( * PFONC )() ; /* déclaration du type pointeur sur */
                /* une fonction renvoyant un entier. */

#define NB_ELTS 5000      /* et allez donc ! */
int tab_global[ NB_ELTS ] ; /* le tableau à trier */

main()
{
  PFONC echange(), compare() ;
  /* Certains C le veulent ainsi ! */

  /* ici remplissage du tableau ou autre... */

  tri_xxx( NB_ELTS, echange, compare ) ;
}

tri_xxx( nb, cmp_cles, swp_elts )
int nb ;
PFONC cmp_cles, swp_elts ;
{
  /* Ici se déroule le tri proprement dit qui pilote les appels */
  /* à ( *cmp_cles )() et ( *swp_elts)() quand il en a besoin. */
}


compare( i, j )
int i, j ;

{
        register int * ptr = ( int * ) tab_global ;

        return( ptr[i] - ptr[j] ) ;
  /* c'est tout car les entiers sont dans ce cas leur propre clé */
  /* si on avait voulu trier par ordre décroissant, il suffisait */
  /* de faire le contraire. */

}

echange( i, j )
register int i, j ;
{
  register int tampon, * ptr = ( int * ) tab_global ;

  tampon = ptr[ i ] ;
  ptr[ i ] = ptr[ j ]
  ptr[ j ] = tampon ;
}
```

Listing 2

arrive!). Une discussion détaillée tri par tri se révèle très utile et fait appel à des notions importantes pour la programmation.

La vitesse

Si vous êtes fachés avec les mathématiques, sautez les parties qui les utilisent mais faites quand même un effort pour ce qui suit. Le critère essentiel qui guide le choix d'un algorithme de tri (ou autre) est sa vitesse d'exécution, il importe donc de se pencher sur le problème général du temps pris en fonction des variables d'entrées. Pour simplifier les choses, considérons que l'exécution de la séquence interne - à l'intérieur des boucles - d'une routine prend une durée fixe t0 qu'on fixera à 1 ms (milliseconde). Prenons pour commencer le cas d'un tableau de mille entiers sur lesquels on veut faire une opération quelconque de durée t0, l'exécution totale prendra une seconde. Pour 10000 éléments on aura évidemment un temps de 10 s et ainsi de suite. On exprime cette relation (linéaire) en disant:

«l'algorithme est de l'ordre de grandeur de n»

ou avec une notation plus brève:

T(n) = O(n).

Mais les choses ne sont pas toujours aussi roses et on démontre qu'un tri à bulles se comporte en fonction des «n» éléments à classer en:

TB(n) = O(n*n)

alors qu'un tri par arbre est en:

TA(n) = O(n*log2(n))

Voyons ce que ceci implique avec la constante t0 et selon différentes valeurs de n:

TB(128) # 1.6 10^4 ms = 1.6 s

TA(128) # 900 ms = .9 s

TB(1024) # 10^6 ms = 1000 s # 15 mn

TA(1024) # 10000 ms = 10 s

Le rapport entre ces deux vitesses d'exécution est éloquant et ne fera que croître et embellir quelle que soit en réalité la constante t0. Il faut tout de même noter deux restrictions:

- Ces valeurs sont des tendances valables pour des nombres assez importants, au moins de quelques centaines. Elles ne sont pas valables en dessous de 50 par exemple.
- Les parties principales n*n et n*log2(n) sont données à une constante multiplicative près qui est inférieure à 1 dans le cas de TB et supérieure à 1 pour TA. Les résultats devraient être tempérés mais ceci ne remet en aucun cas la tendance en cause. Le tri à bulles est définitivement un algorithme pourri qu'il ne servirait à rien d'optimiser ou de coder en assembleur pour gagner quelques pouillèmes quand le nombre de données devient trop important. Insistons sur le fait qu'il faut alors changer d'algorithme et sur l'importance de savoir de quel ordre est celui qu'on utilise!

Il faut tout de même garder à l'esprit qu'un gus qui programme en assembleur et doit classer 10 entiers avec un tri dont il ne se servira plus ensuite serait singulièrement crétin d'aller s'enquiquiner en codant un tri par arbre ou autre quicksort!

Ceci dit, on démontre que le meilleur algorithme possible pour trier un ensemble d'éléments est en :

T(n) = O(n*log2(n))

Ce dont s'approchent, à la fameuse constante multiplicative près, le tri par arbre ou par interclassement ainsi que le tri rapide, etc.

La stabilité

Une fois posé le problème de la vitesse, voici celui de la stabilité. On dit qu'un tri est stable s'il ne modifie l'ordre relatif des éléments de clé égale qu'il rencontre. Il est important d'avoir un tri stable si on veut trier un ensemble avec différents critères successif sans que le dernier classement soit bouleversé par l'opération courante. Un exemple pourrait être la population française classée par taille puis par âge et poids. Dans ce cas, un tri stable est de rigueur. On peut voir par exemple que le tri par insertion est stable alors que le tri shell ne l'est pas.

Les tris de fichiers

Compte tenu de la lenteur de certains périphériques, les tris de fichiers posent quelques problèmes spécifiques liés aux temps d'accès vers un élément donné. La plupart des tris donnés ici ne requièrent aucun espace de stockage intermédiaire alors que, quand on a la

place, il peut être avantageux de créer des fichiers temporaires. La question ne sera pas approfondie ici. Disons toutefois que le tri rapide, le tri par interclassement ou par fusion, sont des compromis acceptables. Précisons encore que dans certains cas, il peut être avantageux de ne pas échanger immédiatement deux éléments, mais de ne faire qu'une affectation en gardant la deuxième sous le coude pour gagner du temps.
Pour bien faire, il faudrait aussi se pencher sur l'espace mémoire que nécessitent certains tris - non traités ici - en utilisant des tableaux intermédiaires de stockage ou même l'encombrement de la pile que peut impliquer une solution récursive comme dans le cas du tri rapide. Sans entrer dans toutes ces finesses - L'encyclopédie de Knuth y consacre bien 300 pages à vue de nez -, voici arrivé le moment de passer en revue les plus connus. Egalement pour simplifier, les explications données se réfèreront, si nécessaire, à un tableau contenant n entiers, indexé à partir de 0 et devant être trié par ordre croissant. Voici donc, - à tout saigneur, toute horreur - qu'apparaît...

...Le tri à bulle

Il fait partie des tris par échanges, c'est-à-dire que l'on va parcourir le tableau en cherchant deux éléments inversés. Si on en trouve on rétablit l'ordre et on relance un nouveau passage jusqu'à ce qu'on ait pu parcourir le tableau sans en trouver d'inversion. Le tableau est alors trié *(listing 3)*.
Un élément mal placé ne va donc remonter dans le tableau que d'une position à la fois. En supposant que le tableau est trié à l'exception du n-k+1ème qui devrait être en premier, il va falloir appeler n-1 fois cmp_cle() à chaque passage dans la boucle for et

```
tri_bulles( nb, cmp_cles, swp_elts )
int nb ;
register PFONC cmp_cles, swp_elts ;

{
    register int i, flag, fin = nb - 1 ;

    if( nb < 2 ) return ;
    do
        for( flag = i = 0 ; i < fin ; i++ )
            if( (*cmp_cles )( i, i+1 ) > 0 )
            {
                flag = 1 ;
                ( *swp_elts )( i, i+1 ) ;
            }
    while( flag ) ;
}
```

Listing 3

```
tri_insert( nb, cmp_cles, swp_elts )
int nb ;
PFONC cmp_cles, swp_elts ;

{
    _tri_insert( 0, nb - 1, cmp_cles, swp_elts ) ;
}

_tri_insert( deb, fin, cmp_cles, swp_elts )
register int deb, fin ;
register PFONC cmp_cles, swp_elts ;

{
    register int i, j ;

    if( fin - deb < 2 ) return ;
    for( i = deb + 1 ; i <= fin ; i++ )
    {
        j = i - 1 ;
        while( j >= 0 && ( (*cmp_cles )( j, j+1 ) > 0 ))
        {
            ( *swp_elts )( j, j+1 ) ;
            --j ;
        }
    }
}
```

Listing 4

comme la remontée est lente, cette boucle for va elle-même être exécutée n-k+1 fois. Bref en n'envisageant que ce cas optimiste et en négligeant les quelques appels à swp_elt(), on trouve:
T(n) = O((n-1)*(n-k+1)) # O(n*n)
A part ça, le tri est stable et se comporte en O(n) si le tableau est déjà trié (k = n).

Le tri par insertion

L'idée de base considère le tableau comme constitué d'un sous-tableau déjà trié de k éléments (au début, k = 1) dans lequel on va insérer l'élément suivant en décalant ceux qui le précèdent jusqu'à ce qu'il trouve sa place et ainsi de suite ad finitum. Ce tri est présenté en deux étapes car il reservira. En effet, tri_rapide aura besoin de _tri_insert() qui peut trier un sous-tableau *(listing 4)*.
En tablant sur le pire des cas, à savoir un tableau complètement inversé, on voit que chaque valeur de i sera suivie de i appels à cmp_cle() et swp_elt() pour ramener l'élément en 1ère position. Comme i varie de 1 à n-1, on aura:

T(n) = O(1 + 2 +...+ n-1) = O(n(n-1) /2) # O(n*n)

Bien qu'étant également en O(n*n), ce tri à une constante multiplicative inférieure à tri bulles. Par des considérations statistiques, (tout tableau n'est pas complètement inversé), on trouve que cette dernière est de l'ordre de 1/4 et dans le cas d'un tableau déjà trié, il se comporte en O(n). Il est donc quatre fois plus rapide que tri bulles. Moralité, si vous désirez un tri facile à programmer, choisissez celui-là.

Le tri par extraction

On trouve ici une application classique de la récursivité, si le tableau [k..n] ne contient qu'un seul nombre, c'est finit. Sinon, on cherche l'élément de clé minimale qu'on échange avec le k ème et on appelle de nouveau la fonction avec le tableau [k+1..n].
On remarque au passage que

```
tri_extract( nb, cmp_cles, swp_elts )
register int nb ;
register PFONC cmp_cles, swp_elts ;

{
    register int i, j, min_ind ;

    if( nb < 2 ) return ;
    for( i = 0 ; i < nb ; i++ )
    {
        min_ind = i ; /* indice de la clé minimale */
        for( j = i + 1 ; j < nb ; j++ )
        {
            if( (*cmp_cles )( min_ind, j ) > 0 )
                min_ind = j ;
        }
        if( min_ind != i )
            ( *swp_elts )( i, min_ind ) ;
    }
}
```

Listing 5

cet algorithme est toujours du type:
T(n) = O(1 + 2 + 3 +...+n-1) = O(n*(n-1) / 2) # O(n*n)

La récursivité s'élimine facilement et on trouve... l'algorithme du listing 5. Sous cette forme, il ne présente aucun intérêt.

Passons aux choses sérieuses...

Bien qu'ils soient intéressants sur le plan des méthodes utilisées, les tris présentés jusqu'à maintenant ne sont pas vraiment praticables en raison de l'insuffisance des algorithmes en O(n*n) dans un domaine ou les données à trier se comptent souvent en milliers.

C'est pourquoi, on a tenté d'améliorer les idées de base pour en tirer des tris en O(n*log2(n)). Ces derniers sont nettement plus compliqués mais ils en valent la peine.

Le tri Shell

Quand on examine le tri par insertion simple, on le trouve très bien sauf que les éléments ne «remontent» que d'une case à la fois vers leurs emplacements définitifs. L'idée vient alors de leur faire «sauter» toutes ces étapes pour gagner du temps. La méthode proposée par Shell consiste à ne pas trier d'un seul coup le tableau mais de le décomposer en plusieurs suites dont chaque élément est espacé d'un certain pas ou incrément et de trier celles-ci par insertion. Dans ce cas, l'échange de deux éléments leur fera franchir 'incrément' places d'un seul coup. Quand on en aura fini avec ces suites, on recommencera avec un incrément plus petit - donc moins de suites et dans un tableau mieux trié - et ainsi de suite jusqua ce qu'il atteigne 1. Cette dernière étape, se ramène d'ailleurs à un tri par insertion, mais dont le tableau aurait déjà été fortement classé. Or on sait que le tri par insertion profite de tout l'ordre qu'il peut trouver. Pour en donner une idée, voici ce qui ce passe avec un tableau de 15 éléments et une suite d'incréments valant successivement 5, 3 et 1... On tri d'abord par insertion les suites d'indices (par pas de 5) :
{ 0, 5, 10 }, { 1, 6, 11 }, { 2, 7, 12 }, { 3, 8, 13 }, { 4, 9, 14 }
puis les suites (par pas de 3) :
{ 0, 3, 6, 9, 12 }, { 1, 4, 7, 10, 13 }, { 2, 5, 8, 14 }
et seulement à la fin la suite (par pas de 1) : { 1, 2, ..., 14 }

Voici donc le programme du tri Shell *(listing 6)* avec la série d'incréments qui valent successivement 1, 4, 13, 40, etc. correspondant à la relation pas(n) = 3 * pas(n-1) + 1).

```
tri_shell( nb, cmp_cles, swp_elts )
int nb ;
register PFONC cmp_cles, swp_elts ;

{
    register int i, j, k, pas ;

    if( nb < 2 ) return ;
    pas = 1 ;
    /* initialisation de la suite */
    while( pas < nb / 9 )
        pas = 3 * pas + 1 ;

    do
    {
        for( i=0 ; i < pas ; i++ )
            /* tri par insertion de la i ème série */
            for( j = i ; j < nb ; j += pas )
            {
                k = j - pas ;
                while( k >= 0 && ( *cmp_cles )( k, k + pas ) > 0 ))
                {
                    ( *swp_elts )( k, k + pas ) ;
                    k -= pas ;
                }
            }
        pas /= 3 ;
    } while( pas >= 1 ) ;
}
```

Listing 6

Ce tri tient bien la route jusqu'à plusieurs milliers (voire dizaine de milliers) d'éléments et son algorithme est donné pour être de l'ordre de 1,5 n^(1.25). Son défaut essentiel qui peut le faire rejeter est de pas être stable (un élément peut sauter par dessus son collègue de même clé placé dans une autre suite). Mais si vous vous en fichez, il présente un bon rapport taille-de-code/rapidité.
Note: Vu l'apparition de micro_ordinateurs possédant une mémoire cache, il faut mettre un bémol aux louanges concernant le tri Shell. En effet, les «sauts» effectués par les éléments à trier peuvent provoquer une réactualisation du cache et donc une baisse des performances.

Le tri rapide

L'idée de base du tri rapide tient en quelque lignes :
a) On choisit un élément qu'on appelle un pivot.
b) On place tous les éléments de clés inférieures à sa droite et ceux de clés supérieures à sa gauche (on ne les trie pas, on se contente de PARTITIONNER le tableau).
c) On ré-appliqùe récursivement la méthode au sous-tableau de droite puis au sous-tableau de gauche.

Prenons un cas favorable en supposant qu'après chaque étape, le pivot sera placé au centre du tableau partitionné. La routine Partition() étant manifestement d'un ordre majoré par n et comme il reste 2 sous-tableaux de n/2 éléments à classer, on aura :
TR(n) <= O(n) + 2 * TR(n / 2) # O(n * log2(n)
Ce qui est de l'ordre de l'optimum théorique. Malheureusement, les choses se gâtent si le choix du pivot ne se porte pas sur un élément proche de la médiane, catastrophe qui se produit dans le cas d'un tableau déjà trié et dans lequel la partition se fait naïvement autour du premier élément des sous-tableaux qui auront alors 1 ou k-1 éléments.
Ce qui donnera :
TR(n) # O(n) + O(1) + T(n-1) # O(n) + O(n-1) +..+O(1) = O(.n*n) ???
On vient en gros de réinventer le tri à bulle! C'est pour éviter une telle dégénérescence que la routine Partition() prend les précautions qu'on voit dans la sélection de son pivot. Mais il faut être bien persuadé que si on peut en rendre la probabilité très faible, elle n'est jamais nulle avec ce tri!
L'autre problème qui peut surgir est celui de l'inutilité de l'espace pris sur la pile pour l'empilement de données concernant des sous-tableaux de quelques éléments. C'est pour cela qu'on fixe une taille minimale, un seuil, qui, s'il est dépassé, déclenche l'appel d'un tri par insertion simple prenant en charge la finition du travail. Cette option présente en outre l'avantage de tirer parti d'un ordre pré-existant déjà dans le tableau. Voici donc la version finale, Rec_sort() est la partie récursive qui exécute la phase c), déclenche le tri par insertion et appelle Partition() pour les phases a) et b) *(listing 7)*.
Le tri rapide est le tri le plus connu et certainement le plus employé. Quand tout va bien (!), c'est aussi le plus rapide. Malheureusement, il présente toujours un risque de dégénérescence. C'est d'ailleurs le critère de choix entre tri_rapide() et tri_arbre() qui va suivre.

Le tri par arbre

Quand on jette un oeil sur le programme du tri par arbre, on se demande ce qui peut bien cacher l'arbre (binaire) dont la structure n'apparaît nulle part. C'est qu'il est en fait contenu implicitement (isomorphisme) dans le tableau à trier. On peut considérer - de façon un peu tordue - que les successeurs gauches et droits d'un élément d'indice

```
tri_rapide( nb, cmp_cles, swp_elts )
int nb ;
PFONC cmp_cles, swp_elts ;
{
    if( nb < 2 ) return ;
    _rec_sort( 0, nb - 1, cmp_cles, swp_elts ) ;
}
_rec_sort( i, j, cmp_cles, swp_elts )
int i, j ;
PFONC cmp_cles, swp_elts ;
{
    int pivot, seuil = 16 ;

    while( ( j - i ) > seuil )
    {
        pivot = partition( i, j, cmp_cles, swp_elts ) ;
        if( ( pivot - i ) <= ( j - pivot ) )
        {
            _rec_sort( i, pivot - 1, cmp_cles, swp_elts ) ;
            i = pivot + 1 ;
        }
        else
        {
            _rec_sort( pivot + 1, j, cmp_cles, swp_elts ) ;
            j = pivot - 1 ;
        }
    }
    _tri_insert( i, j, cmp_cles, swp_elts ) ;
}

partition( i, j, cmp_cles, swp_elts )
register int i, j ;
register PFONC cmp_cles, swp_elts ;

{
    register int u, v ;
    int centre ;

    if( j < i ) return( 1 ) ;

    centre = ( i + j ) >> 1 ; /* division par 2 */
    u = i ;
    v = j + 1 ;
    if( ( *cmp_cles )( j, centre ) < 0 )
      ( *swp_elts )( j, centre ) ;

    if( ( *cmp_cles )( j, i ) < 0 )
      ( *swp_elts )( j, i ) ;

    if( ( *cmp_cles )( i, centre ) < 0 )
      ( *swp_elts )( i, centre ) ;

    while( u < v ) {
        do u++ ; while( ( *cmp_cles )( u, i ) < 0 ) ;
        do v-- ; while( ( *cmp_cles )( v, i ) > 0 ) ;
        ( *swp_elts )( u, v ) ;
    }
    if( u > v ) ( *swp_elts )( u, v ) ;
    u = ( u < v ) ? u : v ;   /* min( u, v ) ; */
    ( *swp_elts )( i, u ) ;
    return( u ) ;
}
```

Listing 7

k seront les éléments d'indices 2k et 2k+1... Une autre propriété intéressante liée à cette manière de voir l'agriculture est que l'arbre - qui pousse vers le bas et de gauche à droite! - va voir tout ses noeuds remplis à l'exception - éventuelle - de ceux placés en bas à droite (faire un dessin), il est donc saturé au possible. Sans rentrer dans les détails, l'idée est dans un premier temps d'obtenir un arbre particulier nommé maximier dont la racine de chaque sous-arbre est l'élément qui a la plus grande clé de tout l'ensemble des éléments composant le sous-arbre. Il est alors facile de pas-

```
tri_arbre( nb, cmp_cles, swp_elts )
register int nb ;
register PFONC cmp_cles, swp_elts ;

{
    register int i ;

    if( nb < 2 ) return ;
    /* Transformer le tableau en une représentation d'un maximier */
    i = nb / 2 ;
    for( ;; )
    {
        if( i < 0 ) break ;
        reorg_arbre( i, nb, cmp_cles, swp_elts ) ;
        i-- ;
    }
    /* Le tri proprement dit */
    i = nb ;
    for( ;; )
    {
        if( i < 0 ) break ;
        ( *swp_elts )( 0, i ) ;
        i-- ;
        reorg_arbre( 0, i, cmp_cles, swp_elts ) ;
    }
}

reorg_arbre( deb, fin, cmp_cles, swp_elts )
register int  fin ;
int deb ;
register PFONC cmp_cles, swp_elts ;

{
    register int right, left, max_ind ;
  int j ;

    j = deb ;
    for( ;; )
    {
        left = 2 * j ;
        if( left > fin ) return ;
        max_ind = left ;
        right = left + 1 ;
        if( right <= fin )
            if( ( *cmp_cles )( left, right ) < 0 )
                max_ind = right ;
        if( ( *cmp_cles )( max_ind, j ) <= 0 )
            return ;
        ( *swp_elts )( j, max_ind ) ;
        j= max_ind ;
    }
}
```

Listing 8

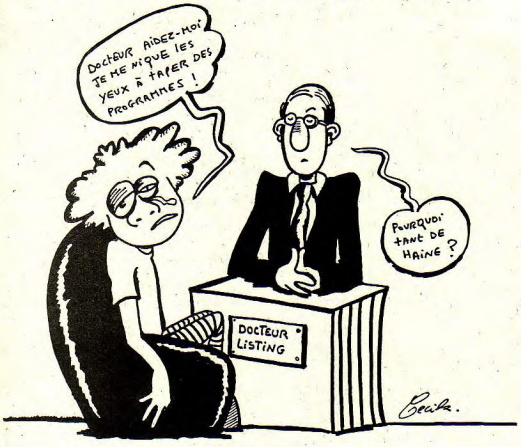
Suite page 63

Un PC musicien!

NOTEDIT (PC carte graphique)

La musique n'est pas le fort de l'ordinateur PC (hormis par le biais des logiciels MIDI). Toutefois, sollicité par un utilitaire musical, sa seule et unique voix ne manque pas de charme.

Ce mini-logiciel de composition musicale en GW Basic, adapté aux possibilités sonores du PC, permet l'écriture de notes sur une portée et leur interprétation.



Utilisation

Après lancement, une série de questions vous sont posées:

• «Nom du fichier»: huit lettres maximum, «PASDENOM.MUS» est retenu par défaut. Répondre «FIL» affiche la liste des fichiers *.MUS présents sur la disquette.

• «Nouveau fichier (O/N)»: «O» fait passer à la question suivante, «N» interprète le fichier appelé et vous demande si vous désirez l'imprimer.

• «Tempo (32/255) par défaut 120»: permet de modifier éventuellement le tempo général. Une simple validation affiche l'éditeur.

Celui-ci présente une portée surmontée des symboles relatifs aux notes, exemple D0=D, RE=R, etc. W affiche une pause.

O: augmente l'octave de 1 et double la valeur de la note.

P: baisse d'un octave et divise par deux la valeur de la note.

G: dièse la note sélectionnée.

A: bémole la note choisie (l'attribution des dièses et des bémols respecte les conventions musicales).

+ : diminue la durée d'une note ou d'une pause. Par exemple de la blanche (2 temps) à la noire (1 temps).

- : augmente la durée d'une note ou d'une pause (maximum 4 temps pour la ronde et minimum 1/16e de temps pour la quadruple croche.

Q : permet de changer un symbole érroné. Frappez Q et inscrivez le nouveau symbole.

Z : fin de saisie, interprétation du morceau.

Exemple (Frère Jacques)

Après lancement, réponse aux questions et affichage de l'éditeur, tapez :

DRMDDRMDMF-S+MF-S++SLSF-MD+SLSF-MDDPSO-D+DPSO-D (Z pour finir).

David Farenzena

```
' • indique l'endroit où
' vous devez frapper Return.
'
1Ø '**** NOTEDIT ***********.
2Ø '**** FARENZENA DAVID ****.
3Ø CLEAR.
4Ø '**** LES ERREURS ****.
```

```
50 ON ERROR GOTO 2070.
60 '**** INITIALISATION DE L'ECRAN ****.
70 SCREEN 2:KEY OFF:WIDTH 80:CLS.
80 '**** DESSIN DE LA CLEF DE SOL ****.
90 SOL$="C1BR23H3U2E2R8F4D6G4L7H7U7E13U5
H1L2G2D37G2L2H".
100 '**** UTILISATION DU POINTEUR ****.
110 LO=4:OC=3:KO=4.
120 PLAY "O="+VARPTR$(OC).
130 PLAY "L="+VARPTR$(LO).
140 GOSUB 1860.
150 '**** ENTREES DES INFORMATIONS ****.
160 LOCATE 9,28:PRINT"(FIL) POUR LA LIST
E DES FICHIERS".
170 LOCATE 20,31:PRINT"
                    ".
180 LOCATE 10,10:INPUT"LE NOM DU FICHIER
 :    ",ZAP$.
190 IF ZAP$="FIL" OR ZAP$="fil" THEN FIL
ES"*.MUS":ZAP$="":GOTO 160 ELSE CLS:GOSU
B 1860.
200 IF ZAP$="" THEN ZAP$="PASDENOM".
210 LOCATE 10,10:INPUT"NOUVEAU FICHIER (
O/N):",ZA$.
220 IF ZA$="N" OR ZA$="n" THEN GOSUB 181
0:PLAY D$.
230 IF ZA$="" THEN 270.
240 LOCATE 10,10:INPUT"L'IMPRIMER      (
O/N):",ZAPO$.
250 IF ZAPO$="O" OR ZAPO$="o" THEN GOSUB
 1910.
260 GOTO 160.
270 LOCATE 10,10:INPUT"LE TEMPO (32/255)
 PAR DEFAUT 120:",TIO.
280 IF TIO=0 THEN TIO=120.
290 IF TIO<32 OR TIO >255 THEN 160.
300 '**** ACTIVATION DU TAMPON ****.
310 PLAY "T="+VARPTR$(TIO).
320 CLS:GOSUB 1860:D$="".
330 B$="MBT"+STR$(TIO)+"O3":GOSUB 1180.
340 '**** ECRAN DE TRAVAIL ****.
350 CLS:GOSUB 1880.
360 LOCATE 2,7:PRINT"DO=D RE=R MI=M FA=F
 SOL=S LA=L SI=C OCT=";OC;"VAL=";LO.
370 LOCATE 3,7:PRINT"DIESE=G BEMOL=A PET
ITE VAL=+ GRANDE VAL=- OC+=O OC-=P"+" T=
";TIO;"FINIR=Z".
380 LOCATE 2,58:PRINT"REPARER=Q PAUSE=W"
.
390 X=30:Y=60:OX=36.
400 '**** DESSIN DE LA PORTEE ***.
410 FOR J=1 TO 4:GOSUB 830:NEXT J:A$=INK
EY$:X=30:OX=36.
420 '**** DEBUT DE LA BOUCLE PRINCIPALE
****.
430 WHILE A$<>"Z" AND A$<>"z".
440 A$=INKEY$.
450 IF A$="D" OR A$="d" THEN GOSUB 900.
460 IF A$="R" OR A$="r" THEN GOSUB 940.
470 IF A$="M" OR A$="m" THEN GOSUB 980.
480 IF A$="F" OR A$="f" THEN GOSUB 1020.
490 IF A$="S" OR A$="s" THEN GOSUB 1060.
500 IF A$="L" OR A$="l" THEN GOSUB 1100.
510 IF A$="C" OR A$="c" THEN GOSUB 1140.
520 IF A$="O" OR A$="o" THEN GOSUB 1400.
530 IF A$="P" OR A$="p" THEN GOSUB 1460.
540 IF A$="+" THEN GOSUB 1580.
550 IF A$="-" THEN GOSUB 1640.
560 IF A$="Q" OR A$="q" THEN GOSUB 1700.
570 IF A$="G" OR A$="g" THEN GOSUB 1940.
580 IF A$="A" OR A$="a" THEN GOSUB 1970.
590 IF A$="W" OR A$="w" THEN GOSUB 2000.
600 IF A$="." THEN GOSUB 2040.
610 IF Y>600 THEN X=X+40:Y=60:OX=OX+40.
620 IF X>150 THEN CLS:GOTO 410.
630 '**** FIN DE LA BOUCLE PRINCIPALE **
**.
640 WEND.
650 '**** SYS. DE MUSIQUE EN FIN DE PROG
RAMME ****.
660 OC=3:LO=4.
670 PLAY "O="+VARPTR$(OC).
680 PLAY "L="+VARPTR$(LO).
690 PLAY "T="+VARPTR$(TIO).
700 PLAY D$.
710 '**** UNE AUTRE? ****.
720 LOCATE 23,10:INPUT "UNE AUTRE (O/N):
",DA$.
730 IF DA$="O" OR DA$="o" THEN 660.
740 '**** L'ENREGISTRER? ****.
750 LOCATE 23,10:INPUT "L'ENREGISTRER (O
/N):",DAV$.
760 IF DAV$="O" OR DAV$="o" THEN GOSUB 1
760.
770 '**** UNE AUTRE? ****.
780 LOCATE 23,10:INPUT "UNE AUTRE      (
O/N):",DAVI$.
790 IF DAVI$="O" OR DAVI$="o" THEN 30.
800 END.
810 '**** SOUS-PROGRAMMES ****.
820 '**** DESSIN DE LA PORTEE ****.
830 PRESET (10,X+15):DRAW SOL$.
840 FOR I=X TO X+20 STEP 5.
850 LINE (10,I)-(630,I).
860 NEXT I.
870 X=X+40.
880 RETURN.
890 '**** DO ****.
900 B$="C".
910 GOSUB 1520:PRESET (Y,X+25):Y=Y+20:GO
SUB 1180:DRAW NO$.
920 RETURN.
930 '**** RE ****.
940 B$="D".
950 GOSUB 1520:PRESET (Y,X+23):Y=Y+20:GO
SUB 1180:DRAW NO$.
960 RETURN.
970 '**** MI ****.
980 B$="E".
990 GOSUB 1520:PRESET (Y,X+20):Y=Y+20:GO
SUB 1180:DRAW NO$.
1000 RETURN.
1010 '**** FA ****.
1020 B$="F".
1030 GOSUB 1520:PRESET (Y,X+18):Y=Y+20:G
OSUB 1180:DRAW NO$.
1040 RETURN.
1050 '**** SOL ****.
1060 B$="G".
1070 GOSUB 1520:PRESET (Y,X+15):Y=Y+20:G
OSUB 1180:DRAW NO$.
1080 RETURN.
```

```
1090 '**** LA ****.
1100 B$="A".
1110 GOSUB 1520:PRESET (Y,X+13):Y=Y+20:G
OSUB 1180:DRAW NO$.
1120 RETURN.
1130 '**** SI ****.
1140 B$="B".
1150 GOSUB 1520:PRESET (Y,X+10):Y=Y+20:G
OSUB 1180:DRAW NO$.
1160 RETURN.
1170 '**** DESSIN DES SYMBOLES ****.
1180 '.
1190 NO$="".
1200 IF LO=1 THEN HG$="PAUSE
   RONDE          ".
1210 IF LO=2 THEN HG$="DEMI PAUSE
   BLANCHE        ".
1220 IF LO=4 THEN HG$="SOUPIR
   NOIRE          ".
1230 IF LO=8 THEN HG$="DEMI SOUPIR
   CROCHE         ".
1240 IF LO=16 THEN HG$="QUART DE SOUPIR
   DOUBLE-CROCHE  ".
1250 IF LO=32 THEN HG$="HUITIEME DE POUP
IR  TRIPLE-CROCHE  ".
1260 IF LO=64 THEN HG$="SEISIEME DE SOUP
IR  QUADRUPLE-CROCHE".
1270 DIE$="C1R6BL3U3D6":BEM$="C1R6".
1280 IF LO=1 THEN NO$="C1E2R4F2G2L4H2":K
I$="C1R8D1L8".
1290 IF LO=2 THEN NO$="C1E2R4F2NU15G2L4H
2":KI$="C1BD2R8D1L8".
1300 IF LO=4 THEN NO$="C1NR8E1NR6E1R4F2N
U15G1NL6G1L4H2":KI$="C1BR8G3L4F8".
1310 IF LO=8 THEN NO$="C1NR8E1NR6E1R4F2N
U15G1NL6G1L4H2BR8BU15F6":KI$="C1F3R4G8".
1320 IF LO=16 THEN NO$="C1NR8E1NR6E1R4F2
NU15G1NL6G1L4H2BR8BU15F6BD3H6":KI$="C1F3
R4G8BE4L4H3".
1330 IF LO=32 THEN NO$="C1NR8E1NR6E1R4F2
NU15G1NL6G1L4H2BR8BU15F6BD3H6BD3F6":KI$=
"C1F3R4G8BE6L4H3BG2F3R4".
1340 IF LO=64 THEN NO$="C1NR8E1NR6E1R4F2
NU15G1NL6G1L4H2BR8BU15F6BD3H6BD3F6D3H6":
KI$="C1F3R4G8BE6L4H3BG2F3R4BG2L4H3".
1350 LOCATE 23,40:PRINT HG$:HG$="".
1360 '**** ECRITURE DES SYMBOLES DANS UN
E CHAINE ****.
1370 C=LEN(B$):C$=LEFT$(B$,C):D$=D$+C$.
1380 RETURN.
1390 '**** UNE OCTAVE EN DESSUS ****.
1400 IF OC<4 THEN Z=18:OC=OC+1:B$=">" EL
SE Z=0.
1410 LOCATE 2,47:PRINT OC.
1420 X=X-Z.
1430 GOSUB 1180.
1440 RETURN.
1450 '**** UNE OCTAVE EN DESSOUS ****.
1460 IF OC>2 THEN Z=18:OC=OC-1:B$="<" EL
SE Z=0.
1470 LOCATE 2,47:PRINT OC.
1480 X=X+Z.
1490 GOSUB 1180.
1500 RETURN.
1510 '**** POUR ENTENDRE LA NOTE JOUEE *
***.
1520 PLAY "O="+VARPTR$(OC).
1530 PLAY "L="+VARPTR$(LO).
1540 PLAY "T="+VARPTR$(TIO).
1550 PLAY B$.
1560 RETURN.
1570 '**** DIMINUER LA VALEUR DES NOTES(
EX DE BLANCHE A NOIRE) ****.
1580 IF LO<64 THEN LO=LO*2.
1590 B$="L"+STR$(LO).
1600 LOCATE 2,54:PRINT LO.
1610 GOSUB 1180.
1620 RETURN.
1630 '**** AUGNENTER LA VALEUR DES NOTES
(EX DE NOIRE A BLANCHE) ***.
1640 IF LO>1 THEN LO=LO/2.
1650 B$="L"+STR$(LO).
1660 LOCATE 2,54:PRINT LO.
1670 GOSUB 1180.
1680 RETURN.
1690 '**** CORIGER UNE NOTE ****.
1700 Y=Y-20.
1710 C=LEN(B$).
1720 F=LEN(D$)-C.
1730 D$=LEFT$(D$,F).
1740 RETURN.
1750 '**** ENREGISTRER LA CHAINE ****.
1760 OPEN "O",#1,ZAP$+".MUS".
1770 WRITE #1,D$.
1780 CLOSE#1.
1790 RETURN.
1800 '**** LECTURE DE LA CHAINE ****.
1810 OPEN "I",#1,ZAP$+".MUS".
1820 INPUT#1,D$.
1830 CLOSE#1.
1840 RETURN.
1850 '**** LA BOITE DU CADRE ****.
1860 LINE(1,1)-(639,199),,B.
1870 LOCATE 3,37:PRINT"NOTEDIT":LOCATE 4
,35:PRINT"----------".
1880 LINE(1,1)-(639,199),,B.
1890 RETURN.
1900 '**** IMPRESSION DE LA CHAINE ****.
1910 LPRINT ZAP$+".MUS",D$.
1920 RETURN.
1930 '**** LES DIESES ****.
1940 IF B$="C" OR B$="D" OR B$="F" OR B$
="G" OR B$="A" THEN B$="+":PRESET (Y,OX+
6):Y=Y+20:GOSUB 1180:DRAW DIE$.
1950 RETURN.
1960 '**** LES BEMOLS ****.
1970 IF B$="D" OR B$="E" OR B$="G" OR B$
="A" OR B$="B" THEN B$="-":PRESET (Y,OX+
6):Y=Y+20:GOSUB 1180:DRAW BEM$.
1980 RETURN.
1990 '**** LES PAUSES ****.
2000 B$="P"+STR$(LO).
2010 PRESET(Y,OX):Y=Y+20:DRAW KI$.
2020 GOSUB 1180.
2030 RETURN.
2040 B$=".".
2050 PRESET (Y,OX+6):Y=Y+20:DRAW"R2":GOS
UB 1180.
2060 RETURN.
2070 IF ERR<25 THEN LOCATE 20,31 :PRINT
"PAS D'IMPRIMANTE !!!!".
2080 RESTORE:RESUME 250.
```

L'enfance de l'art

PUTGET (PC carte graphique)

Cet court utilitaire graphique en GW Basic, outre la réalisation de dessins, permet leur exploitation et animation au sein de programmes.

Pour ce faire, il offre les particularités suivantes:
• Calcul de la dimension (dim) du dessin à animer.
• Enregistrement dans un fichier (.txt) du nom, des coordonnées et de la dimension dudit dessin.
• Impression du fichier «.txt» destiné à la commande DRAW.
• Animation de l'objet afin de juger de son évolution dans l'espace.

Utilisation

Lancez PUTGET sous GWBASIC. Pour le calcul de la dimension du cadre dans lequel figurera le dessin, introduire le nombre de points horizontaux, verticaux et précisez le mode (screen).
- Points horizontaux maximum: 320 en mode 1, 640 en mode 2.
- Points verticaux maximum: 200 pour les deux modes.
Seul le dessin réalisé dans le cadre pointillé pourra être mobile.
L'éditeur utilise les chiffres du clavier: 6 droite, 9 droite/haut, 7 gauche/haut, etc.
Lorsqu'une ligne est tracée, appuyez sur la barre d'espacement afin de sauver les données dans le fichier «.txt». Par exemple, 6 pendant 10 points puis Espace écrira «R10». Soit l'usage des touches suivantes:
B: déplacement sans dessiner.
N: dessiner puis revenir au point de départ.
ENTER: recommencer.
F: voir le dessin avec le système (DRAW).
H: aide mais avec perte du dessin en cours.

Exemple

• Après lancement de PUTGET, répondez N au message «Lire un fichier (O/N)» puisque vous ne désirez pas lire un fichier déjà enregistré.
• Spécifiez 50 points horizontaux et verticaux et choisissez le mode 1.
• Lorsque le rectangle de 50 x 50 apparaît, tenez la touche 6 enfoncée pendant 10 points, puis frappez la touche ESPACE.
• «C1;R10» s'affiche au dessus de l'écran, tapez F.
• Entrez «LIGNE» en réponse au message «Le nom du dessin» et enregistrez-le par O.
• A l'affichage de «L'animer (O/N)», répondez O afin de visualiser l'animation, puis appuyez sur une touche (exemple, ESPACE).
• Répondez O à «Un autre (O/N)».
• De même pour «Lire un fichier (O/N)».
• Ecrivez «LIGNE» comme nom de fichier. Un texte et un dessin (le vôtre) apparaissent.
• Répondez O à «L'imprimer (O/N)» afin d'imprimer votre fichier .TXT.

Emploi du fichier .TXT

Dans vos programmes Basic, un DRAW suivit du fichier txt tracera votre dessin. DIM() calculé par PUTGET réserve la mémoire nécessaire à l'animation. Placez votre sprite entre un GET et un PUT afin de contrôler son mouvement.

```
10 SCREEN 1,0:KEY
OFF:COLOR 8,2:CLS
20 DESSINS="C1;R9F8R35F2R
9F3D3L61H5U10BF12R33H1L
34"
30 DIM O(184)
40 DEF SEG=&HB800: C=1: D
=100:E=C+70:F=D+20
50 PRESET (C,D):DRAW DESS
IN$:GET(C,D)-(E,F),O
60 WHILE A$="":A$=INKEY$
70 PUT(C,D),O:C=C+1
80 IF C>240 THEN C=1
90 PUT (C,D),O
100 WEND
```

David Farenzena

```
' • indique l'endroit où
' vous devez frapper Return.
'
10 '**** PUTGET ***************.
20 '**** FARENZENA DAVID *******.
30 '**** DEFINITION DU MODE ****.
40 SCREEN 1,0:KEY OFF:COLOR 8,2:CLS.
50 '**** ROUTINE D'AIDE ****.
60 GOSUB 1420.
70 ON ERROR GOTO 1540.
80 '**** INITIALISATION DES VARIABLES **
**.
90 CLS:IF C=0 OR D=0 THEN C=130:D=100.
100 GOSUB 1180.
110 '**** SYSTEME DE LECTURE DES FICH. T
XT ****.
120 LOCATE 5,2:INPUT"LIRE UN FICHIER (O/
N):",D$.
130 IF D$="O" OR D$="o" THEN LOCATE 5,2:
INPUT"LE NOM DU FICHIER      :",E$:GOSUB
 970:CLS.
```

```
140 SCREEN 1,0:COLOR 8,2
150 GOSUB 1180
160 CLEAR
170 '**** SYSTEME DE CALCUL DES DIM ****
180 LOCATE 5,2:INPUT"NOMBRE DE POINTS HORIZONTAUX:",A
190 IF A>319 OR A=0 THEN 180
200 LOCATE 7,2:INPUT"NOMBRE DE POINTS VERTICAUX  :",B
210 IF B>199 OR B=0 THEN 200
220 LOCATE 9,2:INPUT"MODE (1/2):",M
230 IF M<1 OR M>2 THEN 220
240 IF M=1 THEN I=2:MO=4:PL=320:LP=2
250 IF M=2 THEN I=1:MO=8:PL=640:LP=1
260 SCREEN M,0
270 OCT=4+INT((A*I+7)/8)*B
280 LOCATE 11,2:PRINT "NOMBRE D'OCTETS :";OCT
290 Q=INT(OCT/2)
300 LOCATE 13,2:PRINT"DIM(";Q;")POUR UN DESSIN FIXE"
310 L=Q MOD MO
320 IF L<MO THEN L=MO
330 R=INT((Q\MO)*MO)+L
340 LOCATE 15,2:PRINT"DIM(";R;")POUR UN DESSIN MOBILE"
350 LOCATE 23,15:PRINT "(UNE TOUCHE)":WHILE INKEY$="":WEND
360 C=INT ((PL-A)/2)
370 D=INT ((200-B)/2)
380 E=C+A:F=D+B
390 '**** DESSIN DU CADRE POINTILLE ****
400 CLS:LINE (C,D)-(E,F),LP,B,&HCCCC
410 '**** INITIALISATION DES VARIABLES ****
420 X=C:Y=D:T=1
430 LOCATE 23,20:PRINT"(F) POUR FINIR"
440 '**** BOUCLE DE DESSIN ****
450 WHILE A$<>"F" AND A$<>"f"
460 A$=INKEY$:P=0:H=0
470 '**** SYSTEME DE DIRECTION ****
480 ON VAL (A$) GOSUB 720,730,740,750,760,770,780,790,800
490 X=X+P:Y=Y+H
500 '**** CONTROLE DU CLAVIER ****
510 IF A$=CHR$(32) THEN GOSUB 840
520 IF A$="B" OR A$="b" THEN GOSUB 820
530 IF A$="N" OR A$="n" THEN GOSUB 1230
540 IF A$="H" OR A$="h" THEN GOSUB 1420
550 IF A$=CHR$(13) THEN GOSUB 940
560 PSET(X,Y),T
570 '**** VALEURES DE X,Y ****
580 LOCATE 23,2:PRINT"X=";X;"Y=";Y
590 '**** FIN DE LA BOUCLE DE DESSIN ****
600 WEND
610 LOCATE 18,2:INPUT"LE NOM DU DESSIN:",D$
620 '**** ECRITURE A L'ECRAN DU DESSIN ****
630 LOCATE 20,2:PRINT CHR$(34):LOCATE 20,3:CLS:PRESET (C,D):DRAW "C1;"+C$
640 LOCATE 1,1:PRINT "C1;"+C$
650 LOCATE 23,3:INPUT"L'ENREGISTRER (O/N)",G$
660 IF G$="O" OR G$="o" THEN GOSUB 1130
670 IF A<310 THEN LOCATE 23,3:INPUT"L'ANIMER (O/N)      :",W$:IF W$="O" OR W$="o" THEN GOSUB 1260
680 LOCATE 23,3:INPUT"UNE AUTRE    (O/N)",J$
690 IF J$="N" OR J$="n" THEN END ELSE GOTO 30
700 END
710 '**** SOUS-PROGRAMMES DE DIRECTION ****
720 P=-1:H=1:J=J+1:B$="G":RETURN
730 P=0:H=1:J=J+1:B$="D":RETURN
740 P=1:H=1:J=J+1:B$="F":RETURN
750 P=-1:H=0:J=J+1:B$="L":RETURN
760 RETURN
770 P=1:H=0:J=J+1:B$="R":RETURN
780 P=-1:H=-1:J=J+1:B$="H":RETURN
790 P=0:H=-1:J=J+1:B$="U":RETURN
800 P=1:H=-1:J=J+1:B$="E":RETURN
810 '**** SOUS-PROGRAMME DU B ****
820 TG=1:T=0:RETURN
830 '**** SOUS-PROGRAMME D'ECRITURE DU DESSIN ****
840 IF TG=1 THEN B$="B"+B$
850 T=1
860 IF RT=1 THEN B$="N"+B$:X=TX:Y=TY
870 Z=LEN(C$)
880 R$=STR$(J)
890 IF J=0 THEN R$=""
900 C$=LEFT$(C$,Z)+B$+R$
910 LOCATE 2,1:PRINT "C1;";C$
920 J=0:B$="":TG=0:RT=0:RETURN
930 '**** NETTOYAGE DU DESSIN ****
940 J=0:X=C:Y=D:C$="":CLS:LINE(C,D)-(E,F),LP,B,&HCCCC
950 RETURN
960 '**** SOUS-PROGRAMME DE LECTURE FICH.TXT ****
970 CLS
980 OPEN"I",#1,E$+".TXT"
990 INPUT#1,C$,R,M
1000 H$="DIM("+STR$(R)+" )"
1010 I$=E$+".TXT"
1020 V$="MODE"+STR$(M)
1030 IF M=2 THEN SCREEN M:WIDTH 80
1040 LOCATE 2,3:PRINT I$,C$,"DIM(";R;")";"MODE";M
1050 CLOSE#1
1060 '**** DESSIN DU FICH.TXT ****
1070 PRESET (C,D):DRAW C$
1080 '**** IMPRESSION DU FICH.TXT ****
1090 LOCATE 23,1:INPUT"L'IMPRIMER (O/N):",F$
1100 IF F$="O" OR F$="o" THEN LPRINT I$,C$,H$,V$
1110 RETURN
1120 '**** CREATION D'UN FICH.TXT(LEURRE) ****
1130 OPEN "O",#1,D$+".TXT"
1140 WRITE#1,"C1;"+C$,R,M
```

```
1150 CLOSE#1.
1160 RETURN.
1170 '**** DESSIN DU CADRE ****.
1180 LINE(1,1)-(319,199),2,B.
1190 LOCATE 2,17:PRINT"PUTGET".
1200 LOCATE 3,17:PRINT"------".
1210 RETURN.
1220 '**** SOUS-PROGRAMME DU N ****.
1230 RT=1:TX=X:TY=Y.
1240 RETURN.
1250 '**** SOUS-PROGRAMME D'ANIMATION DU
 DESSIN ****.
1260 DIM O(R).
1270 DEF SEG=&HB800.
1280 LINE(C-1,D-1)-(E+1,F+1),LP,B,&HCCCC
.
1290 VI=INT(E-C).
1300 DA=(PL-2)-VI.
1310 A$=INKEY$.
1320 LOCATE 23,3:PRINT"UNE TOUCHE !
   ".
1330 PRESET(C,D):DRAW "C1"+C$:GET(C,D)-(
E,F),O.
1340 WHILE A$="".
1350 A$=INKEY$.
1360 PUT(C,D),O:C=C+1.
1370 IF C>DA THEN C=1.
1380 PUT(C,D),O.
1390 WEND.
1400 RETURN.
1410 '**** SOUS-PROGRAMME D'AIDE ****.
1420 CLS:GOSUB 1180.
1430 LOCATE 6,2:PRINT"LES CHIFFRES POUR
DESSINER".
1440 LOCATE 8,2:PRINT"TOUJOURS DANS LE C
ADRE ".
1450 LOCATE 10,2:PRINT"(ESPACE) POUR NOT
ER LA LIGNE".
1460 LOCATE 12,2:PRINT"(ENTER) POUR RECO
MMENCER".
1470 LOCATE 14,2:PRINT"(B) POUR SE DEPLA
CER SANS DESSINER".
1480 LOCATE 16,2:PRINT"(N) POUR DESSINER
 ET REVENIR".
1490 LOCATE 18,2:PRINT"(F) POUR VOIR LE
DESSIN".
1500 LOCATE 20,2:PRINT"(H) POUR DE L'AID
E".
1510 LOCATE 23,15:PRINT"(UNE TOUCHE)".
1520 WHILE INKEY$="":WEND.
1530 CLS: GOSUB 940:RETURN.
1540 IF ERR=25 THEN LOCATE 23,20:PRINT "
PAS D'IMPRIMANTE!!!".
1550 RESTORE:RESUME 1090.
```

Suite de la page 57

ser à la seconde étape qui est le tri proprement dit *(listing 8)*.
S'il se révèle plus lent que le tri rapide, l'avantage du tri par arbre réside dans la certitude qu'on a de ne pas le voir dégénérer en O(n*n). On peut alors majorer avec certitude les temps qui va être pris en termes de l'ordre de O(n*log2(n)).

Conclusion

Pour finir, voici les questions à se poser avant de choisir parmi les algorithmes proposés.

- Combien de données à trier ?
- La stabilité est-elle importante (tri multicritères) ?
- Les données présentent-elles déjà un certain ordre comme quand on ajoute une ou plusieurs fiches dans un fichier préalablement classé?
- Dois-je exécuter mon tri dans un temps maximal donné?

A partir de là, le choix ne devrait poser aucune - ou presque - difficulté. Ceux qui n'auraît pas trouvé leur bonheur dans cet article auront certainement la révélation en consultant l'un des ouvrages cités en bibliographie.

Note : Tri_rapide était le plus rapide - quand tout allait bien ! - jusqu'à l'apparition récente du tri Pidgeon dont je n'ai malheureusement pas l'algorithme et qui en est une amélioration. Après moult polémiques, ce dernier semble avoir été déclaré vainqueur.

Bibliographie

«Méthodes de programmation» par B. Meyer et C. Baudoin chez Eyrolles dans la collection de la D.E.R de l'EDF.
«Programmation avancée» de J.C. Boussard et R. Malh chez Eyrolles.

Citons aussi pour ceux qui veulent souffrir, l'encyclopédie de Knuth en beaucoup de volumes : "The art of computer programming".

Jean-Yves Trétout

Tri	Algorithme	domaine	note
Bulles	n^2	< 200	/
Extraction	n^2	< 200	/
Insertion	n^2	< 500	/
Shell	1.5 n^1.25	< 20000	instable
rapide	n*log2(n)	tous	peut dégénérer en n^2
Arbre	n*log2(n)	> 100	le plus fiable

Pour donner un éléments concret d'appréciation, le tri de 5000 entiers reste inférieur à 5 secondes avec les 3 derniers algorithmes alors qu'il dépasse la minute pour les autres.

Vous êtes au courant?

ALIMENTATION BI-TENSION SIMPLE

D'ordinaire, les articles traitant de petits montages électroniques, restent discrets quant aux alimentations nécessaires...

Nous allons essayer de définir une alimentation facile à mettre en œuvre, capable de fournir deux tensions régulées (5V et 12V) d'un débit de 1A chacune, en utilisant autant que faire se peut les fonds de tiroirs présents chez tout bricoleur en électronique. Bien sûr, il existe actuellement sur le marché une profusion de circuits «étudiés pour»: régulateurs à tension ajustable, à limitation de courant, à régulation thermique, etc...Néanmoins, je vous livre ici le schéma du montage simple que j'utilise habituellement pour alimenter le petit montage du moment. Il s'agit du «régulateur série», connu depuis la nuit des temps électroniques...

Schéma de principe et fonctionnement

Un petit schéma (Figure 1) vaut mieux qu'un long discours, sans toutefois nous dispenser de quelques explications.

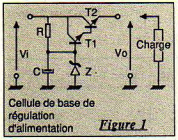


Figure 1

La cellule de régulation proprement dite est constituée de deux composants: la résistance R et la diode Zener Z. Quelle que soit la tension Vi (dans les limites des caractéristiques, bien sûr), la tension aux bornes de la diode Zener sera constante (en réalité, à peu près...). Les transistors T1 et T2 ont pour seul but de fournir un courant important qu'on ne pourrait obtenir avec la diode Zener seule. A noter cependant qu'ils apportent une chute de tension (due au VBE de chacun d'eux) d'environ 1,3 V.

Notre montage

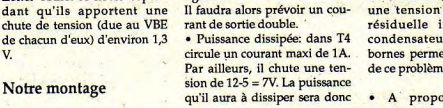


Figure 2

Pour obtenir deux tensions régulées, pourquoi ne pas mettre en série deux circuits de même type? En effet, les 12V du premier sont suffisants, voire redondants pour alimenter le deuxième.

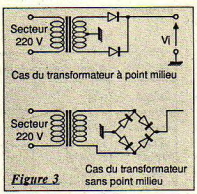


Figure 3

Analyse brève du circuit

Le tranformateur est, sur la figure 2, à point milieu. On peut tout aussi bien utiliser un transfo sans point milieu, en remplaçant les D1 et D2 par un pont redresseur suivant la figure 3.

Il faudra alors prévoir un courant de sortie double.

• Puissance dissipée: dans T4 circule un courant maxi de 1A. Par ailleurs, il chute une tension de 12-5 = 7V. La puissance qu'il aura à dissiper sera donc de 7W au maximum. Par contre, dans T2 circulera un courant maxi de 2A (1A pour le 5V et 1A pour le 12V). Il faut donc réduire la tension d'entrée du montage pour minimiser les pertes dans ce transistor. En choisissant un transformateur délivrant une tension inférieure à 15 Veff, la puissance dissipée dans T2 sera inférieure à 6W. Bien entendu, il faut monter T1 et T2 sur dissipateur pour pouvoir tirer sans risque le maximum de cette alimentation.

• Rendement: des chiffres qui précèdent, on déduit immédiatement que le rendement de l'ensemble avoisine les 50%, ce qui est loin d'être génial. Mais la simplicité et l'efficacité de ce montage compensent largement ce petit inconvénient.

• A propos des condensateurs : quelques mots seulement à propos de C2 et C5. Les diodes Zener sont des régulateurs de tension, mais présentent une impédance non négligeable. Elles sont de ce fait sensibles aux variations du courant qui les traversent, ce qui entraîne une tension de ronflement résiduelle indésirable. Le condensateur placé à leurs bornes permet de s'affranchir de ce problème.

• A propos des Zener: n'oubliez pas qu'une diode ordinaire au silicium présente une chute de tension d'environ 0,6 à 0,7V. Si donc, vous n'avez pas la Zener adéquate, vous pouvez fort bien user du subterfuge suivant (figure 4) :

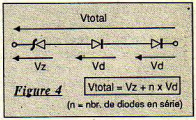


Figure 4

Ceci ne perturbera en rien la régulation et vous permettra d'utiliser le matériel dont vous disposez...

• Sécurité : la protection se fera par l'adjonction d'un fusible (environ 150mA) à l'entrée du transformateur et éventuellement d'un fusible 1,5A sur chacune des sorties (Voir Fu1, Fu2, Fu3).

Michel Hugot