QUICK 55

1ère partie

QUICK 55 vous permet d'accéder à 55 RSXs : ce sont des instructions supplémentaires, comparables à celles du BASIC excepté le fait qu'elles ne résident pas en permanence en mémoire.

La syntaxe est semblable à celle du BASIC : on tape d'abord le nom de l'instruction, SUIVI D'UNE VIRGULE, puis on tape les paramètres nécessaires séparés par une virgule :

ùNNNNNNN, paramètre1, paramètre2,...

Le nom de l'instruction est en fait précédé d'un ù ou | (symbole employé dans l'instruction ùCPM). Un exemple :

ùcircle,0,100,300,200 tracera un cercle de centre de coordonnées (300, 200) et de rayon 100.

Les paramètres pourront en fait être de 2 types :

- Des paramètres d'entrée : c'est le cas dans l'instruction ci-dessus. Ces paramètres permettent la réalisation d'une action (ici le tracé d'un cercle).
- Des paramètres de sortie : ce sont des variables qui, après l'appel d'une RSX, contiendront un résultat. Ici, ce seront des variables de type entier, donc elles s'écriront A%, B%, C%... (le nom suivi de %) ; ces paramètres, avant l'utilisation de la RSX devront impérativement être initialisés : A%=0, B%=0... Les paramètres ainsi initialisés figureront dans les RSX comme suit :

ùNNNNNNN,à A%,à B%,...(la variable entière précédée du @)

Par exemple :
A%=0
ùPRIBUSY,à A%
renverra 0 dans A% si l'imprimante est prête à fonctionner, sinon 1.

* Remarquons enfin qu'une combinaison des 2 types de paramètres est tout à fait possible.

UTILISATION DE POKE ET PEEK

QUICK 55 vous offre l'accès aux 64 K de mémoire supplémentaire dont dispose votre Amstrad ; malheureusement, cette mémoire ne peut être utilisée sous la forme d'un programme BASIC. Elle ne peut servir qu'à stocker des nombres, par le biais des instructions POKE et PEEK :

- POKE adresse, nombre stocke le nombre au niveau de l'adresse ; par exemple poke 27000,200 place le nombre 200 à l'adresse 27000. Il est indispensable que le nombre stocké soit compris entre 0 et 255 ; pour des nombres plus importants, on stocke à l'adresse voulue le reste de la division du nombre par 256 puis au niveau adresse+1 le quotient de cette division. Par exemple, on veut stocker 258 à partir de 27000 :
POKE 27000,2:POKE 27001,1

Pour stocker des chaînes de caractères, il suffira de stocker une succession de codes ASCII.

• PEEK permet de connaître le contenu d'une case mémoire : après avoir tapé le dernier exemple, si vous faites A=PEEK (27000),A contiendra 2 ; il est ainsi possible de reconstituer des chaînes de caractères.

GESTION DE LA MEMOIRE

STRUCTURE DE LA RAM ET LOGEMENT DE QUICK 55

2 possibilités vous sont offertes :

• Vous êtes décidé à faire un programme BASIC de grande envergure : le MEMORY qui précède le lancement de QUICK 55 doit être un MEMORY 39849 : vous disposerez ainsi de 39 K pour votre programme BASIC, mais vous serez dans l'impossibilité d'utiliser les 64 K supplémentaires de la mémoire si votre programme dépasse 16 K.

• Le programme que vous avez décidé de créer ne dépassera pas 16 K : le MEMORY qui précède le lancement de QUICK 55 doit être un MEMORY 16383 : vous pourrez alors utiliser lees 64 K de la mémoire supplémentaire grâce à QUICK 55 pour stocker des images compilées, des SPRITES, ou encore des informations par le biais de PEEK et POKE (cf introduction ou tout autre manuel de référence).

• Toute tentative d'utilisation des 64 K supplémentaires avec un programme BASIC de plus de 16 K entraînera un plantage assuré !!!

Ceci étant clairifié, voici les RSX relatives à la gestion de la mémoire...

• ùCONNECT, (N° de bloc)

avec : n° de bloc compris entre 1 et 5.

Permet de connecter l'un des blocs de mémoire de 16 K (en fait 16,384 K) compris entre 1 et 5 et situés entre 16384 et 32767 : ces blocs de mémoire sont dits "parallèles" : initialement, le bloc n° 1 est connecté ; c'est celui qui est utilisé soit pour le programme BASIC, soit pour stocker des informations, et qui fait partie de la mémoire directement accessible à l'initialisation. Les 4 autres constituent la mémoire supplémentaire (16*4=64 K), sont interchangeables avec le bloc n° 1, mais ne peuvent en aucun cas être utilisés simultanément. Un petit exemple :

```
10 REM EXEMPLE 1
20 FOR I=1 TO 5
30 ùCONNECT,I
   'connecte le bloc I
40 POKE 16384,I:POKE 32767,I
   'I en 16384 et 32767 pour chaque
   bloc
50 NEXT I
60 END
```

Le programme ci-dessus connecte les différents BLOCS de RAM et place au début et à la fin de chacun d'eux leur numéro ; ainsi, après avoir lancé le programme ci-dessus, si vous faites :

ùCONNECT,3:? PEEK (16384):? PEEK (32767)

vous obtiendrez deux fois 3. Comme vous le verrez plus loin, cette instruction est particulièrement utile et est l'un des atouts majeurs de QUICK 55 ; elle simplifie considérablement l'utilisation de la mémoire supplémentaire : le bloc désiré ayant été connecté, il devient aussi accessible que le reste de la RAM ; vous comprenez maintenant qu'on ne puisse utiliser la RAM supplémentaire avec un programme de plus de 16 K ; en effet, un programme de ce type aurait une partie de son listing placé dans le bloc 1, à partir de 16384 K ; un ùconnect,2 déconnecterait (certes sans l'effacer) ce bloc indispensable au bon fonctionnement de votre programme et le remplacerait par un bloc neuf, le bloc 2 : essayez, le plantage est quasiment assuré !!!

• ùTRANSCHAR, (adresse de départ), (premier caractère)

avec : adresse de départ inférieure à 39849

premier caractère étant un code ASCII

Permet de déplacer la table des caractères redéfinissables. Explication : le MEMORY qui précède le lancement de QUICK 55 condamne théoriquement la redéfinition de caractères ; en effet, si vous tentez un SYMBOL AFTER 32, par exemple, vous obtiendrez un Improper Argument. ùTRANSCHAR permet de résoudre ce problème en réallouant une place aux caractères redéfinissables ; vous devez pour cela indiquer une adresse de stockage de la table, ainsi que le code ascii du premier caractère que vous souhaitez redéfinir ; la place mémoire occupée par la table est variable : elle peut aller jusqu'à 2 K si le premier code ASCII choisi est 32 ; il est bien sûr impossible de placer cette table entre 16384 et 32767 si on utilise la mémoire supplémentaire... la table serait en quelque sorte déconnectée et donc inutilisable. Evitez également de la placer en 39000, par exemple, si le premier caractère devant être redéfini est 32 : la table mordrait sur les routines de QUICK 55, et le plantage serait assuré. Il est possible de prévoir la place prise par la table en la plaçant d'abord dans un endroit sûr, puis, à l'aide de PEEKs, en en cherchant la fin (lorsqu'une série suffisamment importante de 0 est trouvée, on peut être assuré que la table s'arrête à l'endroit correspondant ; on soustrait alors cette adresse finale à l'adresse de placement de la table et on obtient la longueur) ; la longueur ayant été calculée, on peut replacer la table à un autre endroit. Un exemple :

Si vous faites ùconnect,37840,32 vous pourrez sans aucun problème faire un SYMBOL 32,..... (le SYMBOL AFTER n'est pas nécessaire) ; mais la mémoire située au-delà de

37840 sera désormais condamnée (s'y trouveront la table suivie des routines de QUICK 55).

• ùTRANSCHARQ,àX%,àY%

avec : X% et Y% deux entiers naturels AYANT ETE DEFINIS AU MOINS UNE FOIS AUPARAVANT PAR X%=0, Y%=0.

Renvoie les paramètres de la routine précédente (le Q qui termine la RSX est le symbole de QUESTION) : supposons qu'auparavant, vous ayez fait un ùTRANSCHAR, 31000,74; tapez les instructions suivantes :

```
A%=0:B%=0
ùTRANSCHARQ,àA%,àB%
? "A% = ";A%
? "B% = ";B%
```

Vous obtiendrez à l'écran A% = 31000 et B% = 74 ; on remarquera que si l'adresse dépasse 32768,A% ne contiendra pas l'adresse mais :

adresse-65536 : un nombre négatif vous sera donc renvoyé, une conversion sera nécesssaire (la valeur d'une variable entière ne peut dépasser 32768).

• ùFILLRAM, (octet remplissage), (adres.depart), (adres.arrivée)

Remplit la zone mémoire qui commence à 'adresse de départ' et se termine à 'adresse d'arrivée par l'octet de remplissage ; si on appelle X et Y ces deux adresses, la RSX est équivalente à :

```
FOR I=X TO Y:POKE I,OCTETremplissage:
NEXT I
```

Cette instruction à l'avantage d'être nettement plus rapide (essayez pour voir) que son équivalent BASIC. Elle est particulièrement utile lorsqu'on décide de nettoyer une partie de la RAM. Par exemple, supposons que vous vouliez nettoyer les 5 blocs de mémoire parallèles de 16 K : il suffit de faire :

```
10 REM EXEMPLE2
20 FOR I=1 TO 5
30 ùCONNECT,I
   'connection du bloc I
40 ùFILLRAM,0,16384,32767
   'nettoyage du bloc connecté
50 NEXT I
60 END
```

• ùTRANS, (adres.depart), (adres. arrivée), (nb octets)

Provoque la recopie de 'nb octets' placés à partir de 'adresse départ' vers la zone débutant à 'adresse arrivée' : en appelant x, y, z ces 3 paramètres, l'équivalent BASIC est :

```
FOR   I=0 TO X-1
POKE  Z+I, PEEK(Y+I)
NEXT  I
```

Là encore, le temps gagné par la RSX est très important. Un exemple d'utilisation : supposons que vous ayez 5 'écrans' enregistrés sur votre disquette (par le classique SAVE "NNNNNN", B,49152, 16384 :

'ECRAN1', 'ECRAN2', 'ECRAN3', 'ECRAN4', 'ECRAN5' ; le programme qui suit permet de charger ces écrans, de les stocker dans la mémoire supplémentaire (rem: cela aurait pu être fait plus simplement en chargeant directement les écrans en 16384, mais le programme qui suit est destiné à expliciter l'utilisation de ùTRANS).

```
10 REM EXEMPLE3
20 FOR I=1 TO 5
30 ùCONNECT,I
   'connection du bloc I
40 A$="ECRAN"+CHR$(48+I)
50 LOAD A$
   'chargement de 'ECRAN I'
   (à partir de 49152)
60 ùTRANS,16384,49152, 16384
   'transfert de l'écran vers bloc I
70 NEXT I
80 END
```

Dans le programme ci-dessus, les blocs mémoire sont connectés un à un ; puis l'écran correspondant est chargé à l'écran, c'est-à-dire à partir de l'adresse 49152 ; de là, l'écran est transféré dans le bloc connecté (à partir 16384) ; on remarque que la longueur du transfert correspond à la longueur de l'écran : 16384 octets.

Cette instruction permet en outre des déplacements de blocs de mémoire particulièrement utiles : transfert de tables de caractères redéfinissables, déplacement d'informations stockées...

• ùCOMP,àX%, (adres.compil), (longueur), (adres.stockage)

avec : X% une variable entière définie à l'avance (X%=0)

Provoque la compilation d'une zone mémoire de taille 'longueur' située à partir de 'adresse de compilation' : la zone compilée est stockée à partir de 'adresse de stockage'. La longueur de la compilation est renvoyée dans X%. Pour ceux qui ne savent pas ce qu'est une compilation, voici quelques explications : supposons que vous ayez à sauvegarder sur disquette une zone de la mémoire :

disons la sauvegarde d'une partie du bloc de ram connecté, de 16384 à 20383 ; cette zone contient inévitablement des séries d'octets identiques ; en particulier, des séries de zéro successifs ; d'où l'idée de regrouper ces octets successifs identiques en deux octets : l'un indiquant le type de l'octet (0,255...), l'autre le nombre de successions de l'octet ; la zone mémoire peut donc par ce moyen être considérablement réduite : dans le cas général, on peut estimer la réduction à une division par deux ; mais cela reste très variable, et comme nous le verrons plus loin, la zone compilée peut même être plus longue que la zone initiale... Il est bien sûr indispensable de conserver la compilation de la zone dans un endroit de la mémoire, pas forcément le même que la zone mémoire elle-même (cela est même à éviter, car des phénomènes de superposition lors de la compilation peuvent se produire) ; il faut d'autre part anticiper la longueur de la compilation : il serait regrettable de prévoir le stockage de la compilation en 38000 alors que celle-ci à toutes les chances de dépasser 2 K... Revenons à notre exemple : on choisit de stocker la compilation en 32768, juste après la fin du bloc connecté (la compilation est supposée ne pas dépasser 7 K). Il suffit de faire pour cela :

```
A%=0
ùCOMP, àA%,16384, 4000, 32768
```

La compilation des octets situés de 16384 à 20383 est maintenant placée à partir de 32768 (la zone mémoire située à partir de 16384 n'a bien sûr pas été altérée) ; la longueur de la compilation, qui, je le rappelle, ne peut être connue à l'avance, est placée en sortie dans A%. Pour sauvegarder la compilation de la zone mémoire concernée, il suffit alors de faire :

```
SAVE "NNNNNN", B, 32768,A%
```

Vous pouvez ensuite rappeler ultérieurement cette compilation sau-

vegardée, qui, dans l'état actuel des choses EST BIEN SUR INUTILISABLE. D'où nécessité de 'décompiler' la zone mémoire : c'est ce que permet l'instruction qui suit...

• ùDECOMP, (adres.stockage), (adres.decompilation)

Provoque la décompilation de la zone mémoire située à 'l'adresse de stockage' ; la zone décompilée est placée à partir de 'adresse de décompilation' ; dans l'exemple précédent, après avoir fait un

LOAD "NNNNNN" (Le bloc compilé est replacé en 32768, sauf si on précise une autre adresse), pour décompiler le bloc, vous devez entrer :

ùDECOMP,32768,16384

Une des principales applications de la compilation/décompilation est la compilation d'écran : l'écran constitue un bloc de 16,384 K (17 K sur le catalogue d'une disquette) ; le nombre d'écrans stockables sur une face de disquette est donc très limité ; d'où l'idée d'utiliser la compilation :

```
10 REM EXEMPLE4:COMPILATION
   D'ECRAN (STOCKAGE PROVISOIRE
   A PARTIR
20 REM DE 16384)
30 A%=0
40 ùCOMP, àA%, 49152, 16384, 16384
   'compilation à partir de 16384
50 SAVE "COMPILE", B, 16384, A%
   'sauvegarde
60 END
```

Pour recharger l'écran, ultérieurement, il vous suffira alors d'intégrer le petit programme suivant :

```
10 REM DECOMPILATION D'ECRAN
   (STOCKAGE PROVISOIRE A PARTIR
   DE
20 REM 20000
30 LOAD "COMPILE", 20000
   'chargement de compile à partir
   de 20000
40 ùDECOMP, 20000, 49152
   'décompilation
40 END
```

Dans le programme ci-dessus, on a choisi de charger le 'COMPILE' à un endroit différent de la compilation, pour bien montrer que la localisation de cette compilation n'a aucune importance ; puis l'écran est décompilé. Quelques remarques ; comme il a été précisé précédemment, la compilation peut à la limite être plus longue que la zone initiale elle-même ; c'est en particulier le cas lorsque vous tentez de compiler une compilation... L'utilisation conjuguée de la mémoire supplémentaire et de ces instructions vous permet donc de garder un nombre d'écrans en mémoire assez important : vous connectez un à un les blocs mémoire parallèles, puis vous compilez dans le bloc connecté autant d'écrans que vous le pouvez ; puis vous passez à un autre bloc, et ainsi de suite.

SYSTEME DISQUE

FORMAT D'UNE DISQUETTE

La disquette sur laquelle vous stockez vos informations dispose d'une organisation interne qui permet de gérer les 178 K disponibles : les 178 K se décomposent en 'pistes' (40) qui elles-même se décomposent en 'secteurs', généralement 9, de 512 octets.

La numérotation des pistes est toujours la même : de 0 à 39 ; mais pour ce qui concerne les secteurs, tout dépend du 'format' choisi au moment du formatage.

- Format système : vous disposez de 9 secteurs par piste, numérotés de 65 à 73 ; inconvénient : les deux premières pistes, 0 et 1 sont réservées à l'installation du CPM.
- Format données : vous disposez encore de 9 secteurs par piste, numérotés de 193 à 201 ; mais cette fois-ci, aucune piste n'est réservée.
- Format IBM : 8 secteurs par pistes numérotés de 0 à 7 (format très peu utilisé).

Comme vous le verrez plus loin, il est possible de stocker des informations sur disquette sans passer par le SAVE habituel : il suffit tout simplement d'écrire sur la disquette une zone mémoire de 512 octets en indiquant la piste et le secteur sur laquelle elle peut être stockée. Voici les RSX concernant l'utilisation du lecteur de disquette :

• ùDISCQ, àX%(0)

avec : X% un tableau unidimensionnel d'entiers naturels ayant été initialisé par : DIM X%(6).

Analyse des paramètres de la disquette insérée, à condition que celle-ci ait été utilisée au moins une fois par l'ordinateur après son insertion (par le biais d'un CAT, d'un SAVE...) ; sinon les paramètres renvoyés dans le tableau X% sont ceux par défaut, ou ceux de la disquette précédemment utilisée. Après envoi de la RSX, vous aurez dans X%(0), X%(1), X%(2), X%(3), X%(4), X%(5), X%(6) les paramètres du système disque :

- X%(0) : contient le numéro du lecteur utilisé : 0 ou 1
- X%(1) : renvoie le numéro USER en cours, c'est-à-dire le numéro de zone du catalogue (pour plus de détail sur USER, consulter ùUSER2).
- X%(2) : contient 255 si aucun fichier n'est actuellement en lecture (fichier ouvert par un OPENIN), sinon renvoie le numéro du lecteur de disquette concerné.
- X%(3) : même principe que pour X%(2), sauf que cette fois-ci, ce sont les fichiers en écriture qui sont concernés (ouverts par un OPENOUT).
- X%(4) : indique le numéro de premier secteur de chaque piste de la disquette (65, 193, ou 0).
- X%(5) : indique le nombre de secteurs par piste (9 ou 8).
- X%(6) : renvoie l'octet de remplissage des secteurs vierges : généralement 229.

Cette instruction est indispensable pour connaître le premier numéro de secteur des pistes, donc pour utiliser les RSX ùRSEC et ùWSEC (voir plus loin). Elle permet également de voir si la lecture ou l'écriture d'un fichier est en cours, et plus spécialement sur quel lecteur de disquette. La connaissance du numéro USER est souvent très intéressante, comme vous allez le voir dans l'instruction qui suit.

• ùUSER2, (numéro user)

avec : numéro USER compris entre 0 et 255.

sélectionne le n° USER. Explication pour ce qui ne connaissent pas les avantages de la fonction USER : la commande BASIC (disponible dès l'initialisation) ùUSER, (numéro USER) avec numéro user compris entre 0 et 15 permet de morceler l'accès aux fichiers contenus sur disquette : initialement,

le numéro USER est 0 ; c'est celui qui apparaît lorsque vous faites la commande CAT. Si maintenant, vous faites ùUSER, 1 puis un CAT, vous obtiendrez un catalogue vide ; en effet, aucun fichier n'a été enregistré alors que le numéro USER était de 1. De même, si vous tentez de lire un fichier enregistré alors que le numéro USER était de 0, vous obtiendrez un message d'erreur ; par contre, si vous enregistrez un programme sous ce numéro USER, il sera invisible et indisponible sous le numéro USER 0. L'intérêt ? Vous pouvez ainsi rassembler les fichiers d'un logiciel sous un même numéro USER, et ainsi séparer les logiciels dans les différents numéros USER. Le problème avec l'instruction BASIC ùUSER, c'est qu'elle ne permet d'avoir accès qu'à 16 USERs. Avec ùUSER2, vous en avez 256 possibles.

Autre avantage de ùUSER2 : vous pouvez récupérer des fichiers effacés par un ùERA : en effet, lorsque vous faites ùERA, "NNNNNN", (s'il reste de la place sur la disquette), le fichier NNNNNN n'est pas directement effacé : il est simplement transféré dans le USER 229. Il suffit donc de faire un ùUSER2,229, et vous avez accès à l'ensemble des fichiers effacés. Vous pouvez alors faire un load, repasser sur le USER 0 et réenregistrer ledit fichier.

• ùERRO, (indicateur)

avec : indicateur égal à 0 ou 255.

Autorise l'affichage des messages d'erreur concernant le disque si l'indicateur est égal à 0, sinon ces messages sont inhibés.

• ùFORMAT, (numéro lecteur), (type format)

avec : numéro de lecteur égal à 0 ou 1 et type de format égal à 65, 193 ou 1.

Sélectionne un type de format :

- type = 65 ⇒ format système.
- type = 193 ⇒ format données.
- type = 1 ⇒ format IBM.

• ùDIRE, (adres. stockage)

Stocke le catalogue de la disquette à partir de l'adresse de stockage' ; une zone de 2,048 K libres suivant l'adresse est nécessaire. L'avantage de cette RSX est qu'elle vous permet d'avoir accès au catalogue de la disquette sans que celui-ci apparaisse à l'écran. LA RSX ayant été exécutée, le catalogue est représenté comme suit (on appelle x l'adresse de stockage) :

- PEEK(X) : 255 pour indiquer la présence du premier fichier
- PEEK(X+1), PEEK(X+2),... PEEK(X+8) : codes ASCII du nom du fichier
- PEEK(X+9), PEEK(X+10), PEEK(X+11) : codes ASCII de l'extension (BAS, BIN,...) avec addition éventuelle de 128 sur chacun de ces octets
- PEEK(X+12) : nombre de K du fichier
- PEEK(X+13) : 0 pour conclure.

- PEEK(X+14) : 255 pour indiquer la présence d'un second fichier..... (répétition de la même série).

Le problème des 128 ajoutés aux codes ascii de l'extension (il s'agit en fait de la validation de certaines options) peut être résolu par le biais de l'instruction AND : A AND 127 ne modifie pas A si A<128 ; Sinon, l'instruction retranche 128 à A :

? 65 and 127 donnera 65.
? 193 and 127 donnera 65.

Notez bien que le stockage du catalogue nécessite 2048 octets : attention aux recouvrements des routines de QUICK 55 !

Pour terminer sur cette instruction, voici un petit programme (incluant le chargement de QUICK 55 qui permet de stocker les noms de fichier du catalogue dans un tableau unidimensionnel (le tableau a été dimensionné à 40 pour ne pas trop prendre de place en mémoire, mais il peut bien sûr être dimensionné au nombre d'entrées maximum du catalogue.

```
1   REM EXEMPLE 5
10  DIM N$(40)
20  MEMORY 16383
30  LOAD "QUICK55"
40  CALL &9BAA:CALL &A1E5:L=1
50  CLS:? "INSEREZ LA DISQUETTE
    SOURCE"
60  C$=INKEY$:IF C$="" THEN 60 ELSE
    ùDIRE,16384
70  FOR I=16384 TO 18432 STEP 14
80  IF PEEK(I) < > 255 THEN 130
90  M$="":FOR J=1 TO 8:M$=M$
    +CHR$(PEEK(I+J)):NEXT J
100 M$=M$+"."
110 FOR K=0 TO 2:M$=M$+CHR$
    (PEEK(I+J+K) AND 127):NEXT K
120 N$(L)=M$:L=L+1
130 NEXT I
```

Le programme ci-dessus dimensionne d'abord le tableau de stockage des noms : N$. Puis "QUICK55" est chargé ; on demande à l'utilisateur de placer la disquette dont il veut avoir le catalogue (attente de la frappe d'une touche) ; le catalogue est alors placé en 16384. Une boucle (I) permet de balayer les 2048 octets du catalogue de 14 en 14 octets : le nom du fichier en cours de formation est stocké dans M$: d'abord le nom en lui-même (boucle J), puis l'extension (boucle K). Le nom final est stocké dans N$(L) ; L est incrémenté. Le ligne 80 permet de vérifier s'il y a effectivement un fichier à partir de I.

• ùHEAD, àX%(0), (adres.stockage), àZ$

avec : Z$ une chaîne de caractères contenant le nom d'un fichier.

X% un tableau unidimensionnel d'entiers naturels ayant été dimensionné par DIM X%(2).

Lecture du header du fichier de nom Z$; l'adresse de stockage doit précéder 2 K libres (tampon de réception) ; en sortie.

- X%(0) contient le type du fichier : 0 si le fichier est un fichier BASIC, 1 si le fichier est BASIC et sauvegardé en protégé, 2 s'il est binaire, 22 s'il est de type séquentiel.

- X%(1) contient l'adresse de début du fichier en mémoire sauf s'il s'agit d'un fichier ASCII : l'adresse du tampon que vous avez proposé est renvoyée.

- X%(2) contient la longueur du fichier (nombre exact d'octets).

Remarquez que l'adresse de stockage et les 2 K libres nécessaires ne sont d'aucune utilité pour vous, mais sont indispensables au bon fonctionnement de la RSX. Un exemple d'utilisation :

```
DIM A%(2)
B$="QUICK55"
ùHEAD,àA%(0), 16384, àB$
```

vous donnera :A%(0)=2
HEX$(A%(1))=&9BAA
A%(2)=2727

Cette instruction vous permet donc de déterminer le type d'un fichier dont l'extension n'est pas une extension ha-

bituelle : pour les programmeurs en assembleur, elle permet de déterminer l'endroit à partir duquel ils peuvent désassembler.

• ùRSEC, (n° de lecteur), (n° de piste), (n° de secteur), (adres.stockage)

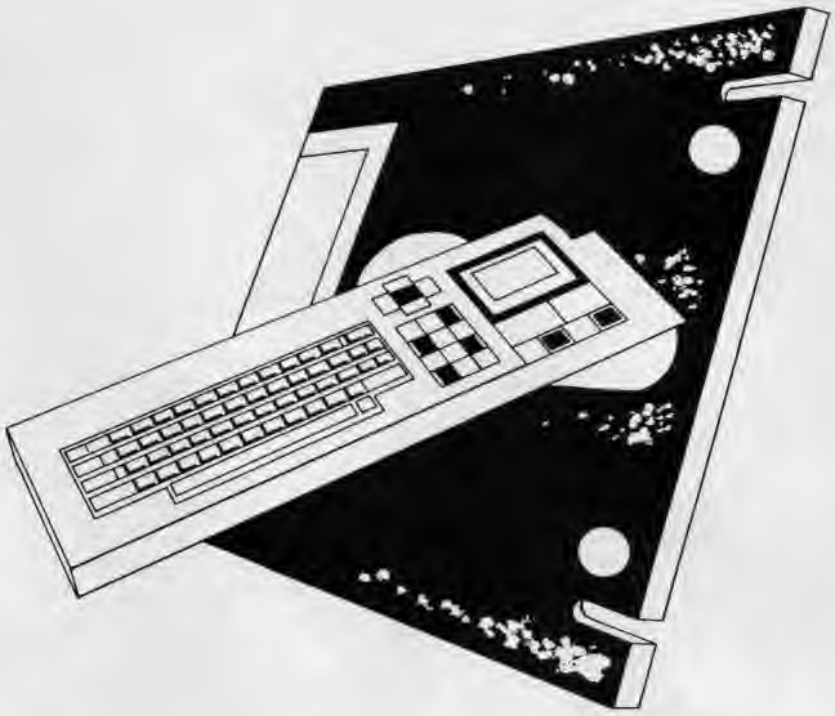
Lecture du secteur 'n° de secteur', situé dans la piste 'n° de piste' ; le numéro du lecteur doit être précisé (0 ou 1) ; le secteur de 512 octets est placé à partir de l'adresse de stockage. Le numéro de secteur dépend du type de format (voir plus haut). Exemple :

ùRSEC, 0, 5, 193, 16384 lit le premier secteur de la piste 5 et le stocke à partir de 16384 (le format est supposé être format données).

• ùWSEC, (n° de lecteur), (n° de piste), (n° de secteur), (adres.stockage)

Effectue l'opération inverse de la précédente : les 512 octets placés à partir de ad.stock sont sauvegardés dans le secteur dont on donne les coordonnées.

Quelques remarques sur ces deux instructions : si vous désirez utiliser les secteurs et que le format de la disquette est inconnu, vous pouvez vous servir de ùDISCQ. Ces deux instructions vous permettent en outre de faire des copies intégrales de disquette : vous chargez au moyen de deux boucles imbriquées les secteurs de chaque piste, et vous les stockez en mémoire ; puis vous changez de disquette et vous les sauvegarder grâce à ùWSEC. Cette opération ne pourra bien sûr pas s'effectuer en une seule fois : vous pouvez cependant utiliser la mémoire supplémentaire pour charger un maximum de secteurs, puis vous les sauvegardez et vous recommencez l'opération avec les secteurs restants. Vous remarquerez dans certains cas que des disquettes n'ont pas un format constant ; en cas d'erreur, changez donc le numéro de secteur. Enfin, n'oubliez pas que l'utilisation des secteurs pour le stockage d'informations vous condamne à ne pas avoir de fichier sur la disquette : en effet, vous ne savez pas (sauf par le biais de logiciels spécialisés) dans quels secteurs est contenu votre fichier ; il risquerait donc d'être détruit. Ceci reste malgré tout un faible inconvénient face aux possibilités énormes qui vous sont offertes par la manipulation des secteurs.



LE CLAVIER

INTRODUCTION

Le clavier constitue l'un des éléments principaux du micro-ordinateur : seul lien entre l'utilisateur et la machine, il est bien souvent source de 'ralentissements' liés à la lenteur du BASIC ; Or, spécialement dans les jeux, il est indispensable. Voici donc quelques RSX qui ont pour principal but de pallier au manque de rapidité du BASIC en ce qui concerne la scrutation du clavier ; quelques autres routines vous permettront simplement d'étendre les possibilités du BASIC dans ce domaine : tests de positionnement de CAPS LOCK...

Vous ne trouverez pas de routines spécialisées AZERTY ou QWERTY, puisqu'actuellement la répartition est à peu près équitable, et qu'il n'y aurait aucune raison valable de privilégier tel ou tel autre type de clavier.
Voici les RSX du clavier...

• ùCLRESET, (indicateur)

avec : indicateur compris entre 0 et 1

Si l'indicateur est à 0 : provoque un simple reset du clavier : réinitialisation des touches de fonction.

Si l'indicateur est à 1 : provoque une réinitialisation totale du clavier : outre le reset précédent, les états et les valeurs de touches, les délais d'attente et de répétition reprennent leurs valeurs par défaut.

Cette routine vous permet par exemple de remettre CAPS LOCK dans son état initial, mais aussi de faire un 'nettoyage' du clavier sans avoir à passer par une succession d'instructions BASIC.

• ùREP, (numéro de touche), (indicateur)

avec : indicateur égal à 0 ou 255

Donne à la touche dont le numéro est fourni la possibilité de se répéter si l'indicateur est à 255, inhibe cette possibilité si l'indicateur est à 0 (cette fonction est également accessible en BASIC par KEY DEF).

• ùREPQ, (numéro de touche), àX%

avec : x% une variable entière initialisée auparavant.

Renvoie 0 dans X% si la touche dont le numéro est fourni ne peut pas se répéter automatiquement, sinon

renvoie 1 dans X%. Cette instruction est la complémentaire de la précédente.

• ùCLWAIT1, àX%

avec : X% une variable entière ayant été initialisée auparavant.

Cette routine provoque l'attente de la frappe d'une touche par l'utilisateur ; en sortie, le code ASCII du caractère frappé est placé dans X% ; cette RSX permet d'éviter le fameux :
1000 C$=INKEY$:IF C$="" THEN 1000 ELSE X%=ASC(C$)

Remarquez que cette attente de frappe n'est pas une attente dynamique : elle convient dans le défilement de menus, par exemple, mais ne peut être utilisée dans le cas ou des actions doivent être exécutées si l'opérateur n'appuie sur aucune touche (jeu d'action...) ; l'ordinateur attend simplement la frappe d'une touche, et, tant qu'une touche ne sera pas frappée, il attendra. D'où nécessité d'utiliser la RSX suivante dans le cas où l'on veut dynamiser l'attente.

• ùCLWAIT2, (code ascii 1), (code ascii 2),..., àX%

avec : X% une variable entière initialisée auparavant.

Cette routine teste si l'une des touches dont le code ASCII est fourni est enfoncée. En sortie, X% contient 0 si aucun caractère n'a été frappé (parmi ceux proposés) sinon renvoie 1 si le premier caractère a été frappé, 2 pour le deuxième, et ainsi de suite. Par exemple, l'instruction ùCLWAIT2, 65, 67, 69, àX% équivaut à :

```
10 C$=INKEY$
20 JF C$="" THEN X%=0
30 IF C$="A" THEN X%=1
40 IF C$="C" THEN X%=2
50 IF C$="E" THEN X%=3
```

Cette RSX vous permet donc d'éviter une belle série de IF THEN, en particulier dans le cas où l'instruction est suivie d'un ON GOSUB. Par exemple, en reprenant le cas précédent, et en supposant qu'un sous-programme doit être effectué si aucune touche n'est frappée :

```
10 X%=0
20 ùCLWAIT2, 65, 67, 69, àX%
30 ON X%+1 GOSUB.....,.....
40 GOTO 20
```

La minimisation du nombre d'instructions permet ainsi de gagner un temps souvent précieux, comme c'est le cas pour les jeux d'action. Notez qu'à la différence de l'instruction précédente, l'instruction est dynamique : la RSX n'attend pas la frappe d'une touche, mais regarde simplement si l'une des touches qui lui est proposées est frappée.

• ùCAPSQ, àX%,àY%

avec : X% et Y% deux variables entières initialisées auparavant.

Contrôle l'état de la touche CAPS LOCK et de la combinaison CONTROL CAPS LOCK ; En sortie :
– X% contient 0 si la combinaison CONTROL CAPS LOCK n'est pas active (verrouillage du clavier pour shift) ; si cette combinaison a été enclenchée, X% contiendra 255 en sortie.
– Y% contient 0 si CAPS LOCK est dans son état par défaut (majuscules seulement accessibles avec SHIFT ou verrouillage du clavier par CONTROL CAPS LOCK), contient 255 si CAPS LOCK a été enfoncé.

Notez que les deux variables renvoient des résultats indépendants : X% peut très bien donner 255 alors que Y% contient 0 : ce n'est pas parce que CONTROL CAPS LOCK a été enfoncé que CAPS LOCK seul l'a été.

• ùCAPS, (état CTRL CAPS LOCK), (état CAPS LOCK)

Cette instruction permet d'agir sur les résultats de la précédente :

– Si 'état CTRL CAPS LOCK' vaut 0, la combinaison CONTROL CAPS LOCK est désactivée ; au contraire, si la valeur envoyée est 255, la combinaison est activée.

– Si 'état CAPS LOCK' vaut 0, on repasse en mode minuscules (le clavier peut cependant être verrouillé par l'indicateur ci-dessus) ; s'il vaut 255, on passe en mode majuscules.

Par exemple : ùCAPS, 0, 255 fait passer en mode majuscules et déverrouille le clavier. Cette RSX permet donc d'éviter les UPPER$... en agissant directement sur le clavier : vous serez ainsi sûr que ce sont des majuscules ou des minuscules qui sont frappées.

L'ECRAN

INTRODUCTION

L'écran est l'un des éléments centraux de l'ordinateur ; pourtant, la programmation de la mémoire vidéo sur CPC est loin d'être aisée, d'autant moins que la tâche du programmeur n'a pas été facilitée : une organisation de la ram écran à passer des nuits blanches, absence de sprites, de scrollings... Bref, il y a beaucoup à faire, et les routines qui vont suivre ont pour but de vous simplifier la programmation.

Vous allez trouver dans les pages qui suivent de nombreuses routines 'd'analyse' de l'écran : elles vous permettront par exemple de savoir en quel mode vous êtes, de connaître le numéro de PAPER utilisé... Cela peut paraître paradoxal de proposer des routines qui vous signalent que vous êtes en MODE 1 alors que vous avez tapez quelques lignes plus haut MODE 1 ; ces routines sont en fait destinées à être utilisées dans le cas d'une manipulation d'un écran extérieur au logiciel que vous avez créé : à priori, le logiciel ne connaît pas le mode, les encres de l'écran sur lequel il va avoir à travailler, et ces routines lui donnent des informations souvent indispensables.

De nombreuses instructions, telles que SCROLL2 utilisent des coordonnées texte (ligne, colonne) ; même si vous travaillez en mode 0, ces coordonnées sont celles du mode 2 (25 lignes, 80 colonnes).

Voici les RSX de l'écran :

• ùSRESET

Provoque une réinitialisation de l'écran : le MODE 1 est restitué avec les couleurs par défaut ; il s'agit à proprement parler d'un nettoyage radical, la base écran étant remise en &c000.

• ùSINIT

Initialisation du mode texte : encres, fenêtres, mode opaque reprennent leur valeurs ou position par défaut.

• **ùMODQ, àX%**

avec : X% une variable entière initialisée auparavant.

Renvoie le mode de l'écran dans X% : 0, 1 ou 2.

• **ùWINDQ, àX%, àY%, àZ%, àT%**

avec : X%, Y%, Z%, T% des variables entières initialisées auparavant.

Envoie dans X%, Y%, Z%, T% les limites de la fenêtre courante : dans X% et Y% les coordonnées du coin en haut à gauche, dans Z% et T% les coordonnées du coin en bas à droite.

• **ùINKQ, X, àY%, àZ%**

avec : Y% et Z% deux variables entières initialisées auparavant.

Permet d'accéder aux paramètres de couleur de l'INK dont le numéro est fourni dans X : en sortie, dans Y% et Z%, sont placées les deux numéros de couleur (le second numéro correspond à la couleur qui apparaît en alternance avec la première lors d'un affichage clignotant).

• **ùPENQ, àX%**

avec X% une variable entière initialisée auparavant.

Renvoie dans X% le numéro du PEN en cours.

• **ùPAPERQ, àX%**

avec X% une variable entière initialisée auparavant.

Renvoie dans X% le numéro de PAPER en cours.

• **ùOPAQ, àX%**

avec X% une variable entière initialisée auparavant.

Renvoie 0 DANS X% si l'on est en mode OPAQUE, sinon renvoie 1.

• **ùSPINKQ, àX%, àY%**

avec X% et Y% des variables entières initialisées auparavant.

Renvoie dans X% et Y% les deux paramètres de l'instruction SPEED INK correspondant aux périodes d'affichage des deux couleurs composant les encres.

• **ùBORDERQ, àX%, àY%**

avec X% et Y% deux variables entières initialisées auparavant.

Renvoie dans X% et Y% les deux paramètres de la commande BORDER, c'est-à-dire les deux couleurs qui s'affichent en alternance.

• **ùLIMITQ, àX%, àY%**

avec X% et Y% deux variables entières initialisées auparavant.

Renvoie dans X% et Y% les limites de l'écran ; cette instruction, peut-être la plus importante de toute cette série, permet non seulement de déterminer le mode mais permet en fait de connaître les limites de l'écran, qui diffèrent selon que l'on est en mode 0, 1 ou 2. Par exemple, si l'on est en mode 2, X% et Y% contiendront respectivement 80 et 25. Cette RSX vous permet donc, par exemple dans le cas de tests liés à la dimension de l'écran, de vérifier si tel ou tel objet se trouve dans les limites correctes de l'écran, sans pour cela avoir à connaître le mode dans lequel le programme se trouve.

• **ùCHON, (indicateur)**

Inhibe l'affichage des caractères si l'indicateur est 0 : si l'indicateur est à 1, l'affichage des caractères est autorisé. C'est cette RSX qui est utilisé directement en assembleur pour la RSX ùDIRE: l'affichage des caractères est inhibé, le catalogue est alors uniquement stocké en mémoire ; puis l'affichage des caractères est à nouveau autorisé. La routine permet donc d'effectuer des opérations liées à l'affichage des caractères (c'est le cas de l'instruction CAT), sans pour cela que l'utilisateur du logiciel en soit averti.

• **ùNOIR**

Provoque un 'flash' à l'écran : à utiliser dans les jeux pour les effets spéciaux.

• **ùSCROLL1, (direction)**

Provoque un scrolling de l'ensemble de l'écran, dans la direction indiquée par le paramètre :

'direction'=1 ⇒ scrolling vers la gauche.
'direction'=2 ⇒ scrolling vers la droite.
'direction'=3 ⇒ scrolling vers le haut.
'direction'=4 ⇒ scrolling vers le bas.

Cette instruction permet ainsi d'éviter de longues procédures BASIC (ce qui est non négligeable dans un jeu d'action) pour déplacer ce qui apparaît à l'écran. Quelques remarques : la partie de l'écran qui disparaît réapparaît à l'autre bout après scrolling ; toutefois, le scrolling horizontal provoque le déplacement des caractères effacés une ligne plus bas, à l'autre bout de l'écran. D'où un décalage de l'ensemble. Voici un petit exemple pour fixer les idées :

```
10 MEMORY 16383
20 LOAD «QUICK55.BIN»
30 CALL &9BAA:CALL &A1E5
40 A%=0
50 ùCLWAIT2,242,243,240,241,àA%
60 IF A%=0 THEN 50 ELSE
   ùSCROLL1,A%:GOTO 50
70 END
```

Ce programme, après le chargement de QUICK55, et initialisation de la variable A%, effectue une attente dynamique de touches : les codes ASCII correspondent aux codes des touches du déplacement curseur. Dès que l'une des touches est enfoncée, le scrolling correspondant de l'écran est effectué ; puis on retourne à l'attente touche.

Ce type de scrolling n'est cependant pas idéal pour être utilisé dans un jeu, du fait des problèmes liés aux remplacements des caractères décalés. D'où la nécessité de l'instruction qui suit.

• **ùSCROLL2, (direction), (co1), (co2), (co3), (co4), (octet remplissage)**

Provoque un scrolling d'une partie de l'écran ; deux directions sont possibles pour le scrolling : vers le haut (direction=1) ou vers le bas (direction=0). La fenêtre qui doit défiler est définie par les coordonnées du coin en haut à gauche (co1 et co2) et par les coordonnées du coin en bas à droite (co3 et co4), ces coordonnées étant bien sûr comprises dans l'écran TEXTE. Ici, la ligne texte qui disparaît ne réapparaîtra pas à l'autre bout de la fenêtre mais la ligne libérée par le décalage sera remplie avec 'octet remplissage'. Il est bien sûr possible de prévoir l'apparition de nouveaux caractères qui viendront sur la place libérée : on retrouve en fait le principe du jeu IKARI WARRIOR. Ce type de scrolling, certes plus difficile à manier

que le précédent, me paraît cependant plus puissant, dans la mesure où l'on sait exactement ce qui va disparaître, ce qui va réapparaître. Voici un petit exemple d'application de SCROLL2 :

```
10  MODE 2
20  MEMORY 16383:LOAD "QUICK55.
    BIN":CALL &9BAA:CALL &A1E5
30  FOR I=1 TO 14 : ùSCROLL2, 1, 10,
    10, 62, 18, 0
40  LOCATE 17, 18:READ A$:PRINT
    A$:NEXT I
50  RESTORE:GOTO 30
100 DATA **************************
110 DATA **                      **
120 DATA **                      **
130 DATA **       QUICK 55       **
140 DATA **                      **
150 DATA **    SCROLL2 vous      **
160 DATA **     permet de        **
170 DATA **     dynamiser        **
180 DATA **   l'affichage de     **
190 DATA **    notices, textes   **
200 DATA **   en tous genres.    **
210 DATA **                      **
220 DATA **                      **
230 DATA **************************
```

Dans le programme ci-dessus, après chaque scrolling, une nouvelle ligne de DATAS est lue et placée à l'endroit 'nettoyé' par l'octet 0 qui a été choisi dans la commande SCROLL2. Dans de tels types de programmes, un réglage de la vitesse de défilement est bien sûr indispensable (boucle d'attente ou commande ùTIME, voir plus loin. L'avantage d'un vez faire défiler juste une partie de l'écran, le reste étant fixe (scores dans un jeu,..).

• ùSPRITE2, (adres.stock), (co1), (co2), (co3), (co4)

Voici la première instruction concernant les sprites : cette RSX 'photographie' une partie de l'écran et la stocke en mémoire ; la zone écran est stockée à partir de 'adresse stockage'. La partie de l'écran stockée est donnée par co1, co2, co3 et co4, ces 4 nombres étant les coordonnées dans le mode texte 2 du rectangle-écran stocké : co1 la ligne du coin en haut à gauche, co2 la colonne de ce même coin ; co3 le nombre de colonnes du sprite et co4 le nombre de lignes. La place prise en mémoire par le SPRITE est donnée par : co3*co4*8.

ùSPRITE2, 21000, 1, 10, 13, 15

Le rectangle affiché à l'écran (qui peut contenir aussi bien des dessins que des caractères), donné par (1, 10) pour le coin en haut à gauche, de longueur 13 et de hauteur 15, sera stocké à partir de 21000. Vous pouvez ainsi stocker plusieurs sprites en mémoire, et les rappeler successivement avec l'instruction qui suit.

• ùSPRITE1, (adres.stock), (co1), (co2), (co3), (co4)

Cette instruction effectue l'opération inverse de la précédente, et replace à l'écran un sprite stocké en mémoire : c'est l'adresse de stockage qui est donnée en premier, suivie des coordonnées du sprite. Les coordonnées doivent bien sûr aboutir à un rectangle de même taille que le rectangle qui avait été stocké, sinon des problèmes surviendront automatiquement : Chaque SPRITE1 qui pointe sur une zone mémoire doit inévitablement avoir été précédé d'un SPRITE2 pointant sur cette même zone mémoire et ayant stocké un rectangle d'octets de même dimension.

Cette instruction peut facilement être combinée avec la précédente pour le déplacement d'objets ou personnages à l'écran ; voici le type d'algorithme à employer :

- stockage de l'objet dessiné en mémoire (à partir de AD1).
- affichage d'un décor.
- stockage provisoire de la partie du décor qui va être effacée par l'objet (à partir de AD2).
- affichage de l'objet à l'endroit correspondant (avec SPRITE1, à partir de AD1).
- l'objet se déplace : la partie du décor provisoirement effacée est réaffichée sur l'objet (avec SPRITE1, à partir de AD2). Puis l'ensemble des opérations se répète à côté, pour simuler le déplacement.

Vous pouvez ainsi multiplier le nombre d'animations, en utilisant au besoin les 64 K supplémentaires de la mémoire. Les sprites vous permettront également de réaliser des programmes de mise en page, des menus déroulants en suraffichage.... L'éventail des possibilités est assez large.

• ùFILLC, (co1), (co2), (co3), (co4), (octet remplissage)

Remplit une zone de l'écran avec un type d'octet : l'octet remplissage ; en particulier, si vous choisissez 0, l'instruction provoque un nettoyage de la partie de l'écran concerné. La zone écran concernée est donnée par des coordonnées de type texte (ligne, colonne) ; co1 et co2 donnent la ligne et la colonne du coin supérieur gauche, co3 et co4 la ligne et la colonne du coin inférieur droit. De même que précédemment, les coordonnées sont celles du mode 2, même si vous vous trouvez en mode 0. L'octet choisi est naturellement compris entre 0 et 255. Cette instruction vous permet, outre des 'nettoyages' de réaliser par exemple des histogrammes ; vous indiquez la taille de votre colonne histogramme, et celle-ci sera remplie avec l'octet choisi.

• ùFILLO (adres.depart), (co1), (co2), (octet remplissage)

Cette instruction est semblable à la précédente, à la différence près que les coordonnées ne sont plus des coordonnées texte mais des adresses mémoires (davantage réservée aux connaissseurs de la mémoire vidéo) : l'adresse départ est l'adresse du premier octet à l'écran qui va être changé en octet de remplissage ; il correspond au coin supérieur gauche ; co1 et co2 donnent la longueur et la hauteur du rectangle devant être rempli, cette fois-ci en terme d'octets (80 disponibles en longueur, 200 en hauteur). L'avantage par rapport à la RSX précédente est la précision gagnée en hauteur.

• ùSOCTET, (adres.stock), (co1), (co2)

Renvoie dans adres.stock est adres.stock+1 l'adresse de l'octet correspondant au caractère dont les coordonnées sont données par co1 et co2 (coordonnées en mode 2). L'adresse de l'octet est :

(contenu de adres.stock+1)*256+ (contenu de adres.stock)

Cette routine sera particulièrement utile pour se servir de ùFILLO. Toutefois, de même que FILLO, cette instruction est à réserver à ceux qui ont des connaissances de base sur la mémoire vidéo.

LES GRAPHIQUES

INTRODUCTION

Voici l'avant-dernier groupe d'instructions de QUICK55 : les routines qui suivent vous proposent :
- d'analyser les paramètres graphiques (coordonnées de la fenêtre graphique...)
- d'améliorer la vitesse de tracé de vos graphiques, tant sur le plan programmation que sur le plan vitesse d'exécution.

Comme vous pourrez le constater, le gain de temps d'exécution pour le tracé du cercle est assez important ; pour des tracés du type triangle, rectangle, le gain de temps est plutôt du côté de la programmation.

Voici les RSX graphiques :

• ùGINIT

Provoque une réinitialisation des graphiques : l'encre graphique est remise à 1, celle du fond à 0 ; l'origine est remise à (0, 0) ; la fenêtre graphique est annulée.

• ùORIQ, àX%, àY%

avec : X% et Y% deux variables entières initialisées auparavant.

Renvoie dans X% et Y% les coordonnées physiques de l'origine (celles définies par ORIGIN).

• ùWGRAQ, àX%, àY%, àZ%, àT%

avec : X%, Y%, Z% et T% des variables entières initialisées auparavant.

Renvoie dans X%, Y%, Z%, T% les coordonnées de la fenêtre graphique : dans X% et Y% les coordonnées du sommet gauche, dans Z% et T% les coordonnées du coin en bas à droite.

• ùGPENQ, àX%

avec : X% une variable entière initialisée auparavant.

Renvoie dans X% le numéro de GRAPHICS PEN.

• ùGPAPERQ, àX%

avec : X% une variable entière initialisée auparavant.

Renvoie dans X% le numéro de GRAPHICS PAPER.

• ùRECT, (indic), (co1), (co2), (co3), (co4)

Trace un rectangle défini par les coordonnées du sommet gauche : co1 et co2 ; sa longueur : co3 et sa hauteur : co4. Si l'indicateur est à 1, le rectangle tracé est un rectangle plein ; si cet indicateur est à 0, le rectangle tracé est un rectangle simple. Par exemple :

ùRECT, 1, 10, 10, 10, 1000, 50

trace un rectangle plein ; les coordonnées du sommet gauche sont 10 et 10 ; la longueur est de 100, la hauteur est de 50 ; les coordonnées sont les coordonnées normales, celles qui se rapportent à l'origine.

• ùTRIANG, (co1), (co2), (co3), (co4), (co5), (co6)

Trace un triangle défini par les coordonnées des 3 sommets : (co1, co2), (co3, co4) et (co5, co6). Les coordonnées sont celles qui se rapportent à l'origine. Par exemple :

ùTRIANG, 10, 10, 100, 100, 300, 200

• ùCIRCLE, (indic), (rayon), (co1), (co2)

Trace un cercle défini par le rayon et les coordonnées du centre (co1 et co2). Si l'indicateur est à 1, le cercle tracé est un cercle plein, sinon le cercle est simple. Les coordonnées du centre, contrairement aux instructions précédentes, sont les coordonnées physiques, qui ne tiennent pas compte d'un éventuel changement d'origine. Par exemple, supposons que vous ayez fait un ORIGIN -500, -500, si vous faites :

ùCIRCLE, 1, 50, 320, 200

vous obtiendrez un cercle plein qui s'affichera au centre de l'écran ; ceci s'explique par le fait que lors du tracé du cercle, pour faciliter le travail, un changement d'origine aux coordonnées du centre est effectué ; ce changement d'origine se fait selon les coordonnées physiques, indépendamment de l'ancienne origine.

• ùSOCTET2, (adres.stock), (co1), (co2)

Renvoie dans adres.stock et adres.stock+1 l'adresse de l'octet sur lequel se situe le point dont les coordonnées ont été placées dans co1 et co2 : pour plus de détail, voir SOCTET, qui est basé sur le même principe, à la différence près qu'ici les coordonnées envoyées sont des coordonnées graphiques et non pas texte ; à réserver aux habitués de la programmation de la mémoire vidéo.

LANCEMENT DE QUICK 55

Introduisez la disquette contenant QUICK55.BIN et faites :

MEMORY 16383 (ou toute autre adresse inférieure à 39850)
LOAD «QUICK55»
CALL &9BAA
CALL &A1E5

Ces instructions indispensables peuvent être soit frappées directement, soit insérées dans un programme BASIC. Les 2 CALL sont très importants : ils permettent tout simplement d'initialiser le logiciel. Ceci fait, vous allez pouvoir utiliser QUICK 55.

COMPLEMENTS

INTRODUCTION

Voici les dernières instructions de QUICK55 : une routine d'attente, une pour la réinitialisation du son, deux pour l'imprimante.

• ùTIME, (temps d'attente)

Provoque une attente correspondant au temps demandé ; permet d'éviter le traditionnel

FOR I=1 TO ...:NEXT I

Une attente de 120 correspond à peu près à 10s.

• ùSDRESET

Provoque une réinitialisation complète des sons : tous les sons de tous les canaux sont éliminés, les files d'attente vidées ; mais les enveloppes sont conservées.

• ùPRIRESET

Provoque un reset de la sortie parallèle, celle de l'imprimante.

• ùPRIBUSY, àX%

avec X% une variable entière initialisée auparavant.

Regarde si l'imprimante est prête (allumée, papier présent...) : en sortie, X% vaut 0 si l'imprimante est prête, sinon vaut 1.

Jean Louis Benard

QUICK 55 a été réalisé en assembleur, et les 55 instructions supplémentaires qu'il représente ne tiennent que sur à peine 3 K de mémoire. Ceci n'a été possible qu'au prix d'un certain sacrifice, que les programmeurs comprendront : les instructions ne protègent pas d'un mauvais passage de paramètres et une instruction mal utilisée peut provoquer un blocage de la machine. Il est donc vivement recommandé de sauvegarder les programmes utilisant QUICK 55 avant de les lancer.
En outre, il est conseillé de sauvegarder QUICK 55 sur une disquette à part : une manipulation des secteurs sur la disquette contenant les RSX pourrait effacer celles-ci. D'ailleurs, l'idéal est de sauvegarder QUICK 55 sur chaque disquette contenant un logiciel qui utilise les RSX : leur chargement à partir de votre programme sera ainsi facilité. Pour cela, il suffit de faire :

MEMORY 16383:LOAD "QUICK55.BIN"

puis d'introduire la nouvelle disquette et de taper :

SAVE "QUICK55",B,&9BAA,2728

QUICK 55 sera ainsi transféré sur la disquette contenant votre logiciel.

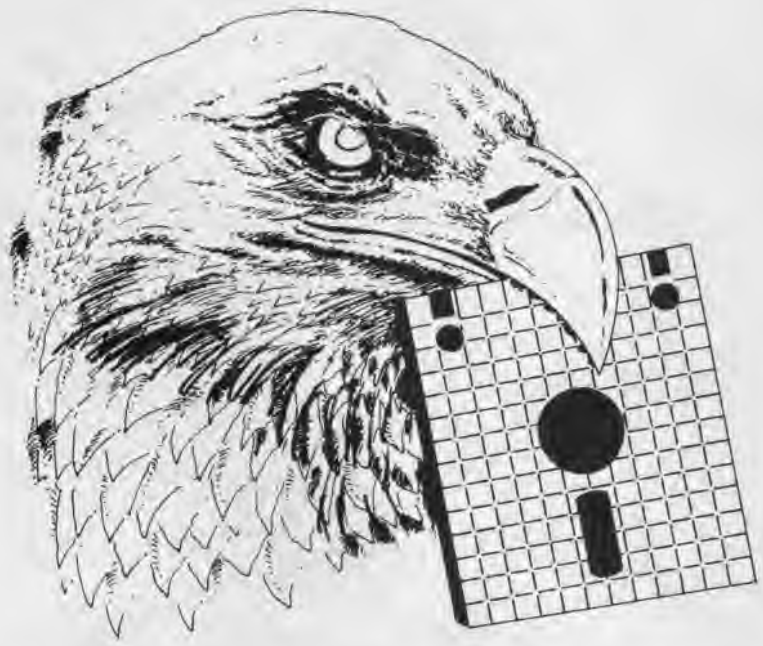
QUICK.BAS

```
10 A=&9BAA:F=&A652:L=100:WHILE A<=F:FOR A=A TO A+15:READ C$:K=
VAL("&"+C$):S=S+K+65536*(S+K)32767):IF A<=F THEN POKE A,K
20 NEXT:READ D$:T=VAL("&"+D$):IF T<>S THEN PRINT CHR$(7);"Erre
ur ligne";L:END ELSE L=L+5:WEND
30 SAVE "quick55",b,&9BAA,2728
100 DATA 01,B3,9B,21,14,9D,C3,D1,BC,30,9C,C3,18,9D,C3,4D,07C5
105 DATA 9D,C3,69,9D,C3,7E,9D,C3,8D,9D,C3,AE,9D,C3,BD,9D,1221
110 DATA C3,D0,9D,C3,04,9E,C3,26,9E,C3,5E,9E,C3,A3,9E,C3,1BC3
115 DATA BF,9E,C3,B7,9E,C3,DD,9E,C3,ED,9E,C3,23,9F,C3,41,264D
120 DATA 9F,C3,4D,9F,C3,51,9F,C3,55,9F,C3,9C,9F,C3,AB,9F,3010
125 DATA C3,BA,9F,C3,CF,9F,C3,DE,9F,C3,03,A0,C3,25,A0,C3,3A4E
130 DATA 48,A0,C3,32,9E,C3,3C,9E,C3,51,A0,C3,60,A0,C3,99,4339
135 DATA A0,C3,B3,A0,C3,40,A1,C3,49,A1,C3,60,A1,C3,73,A1,4CDB
140 DATA C3,7F,A1,C3,97,A1,43,4F,4E,4E,45,43,D4,46,49,4C,541E
145 DATA 4C,52,41,CD,54,52,41,4E,D3,54,52,41,4E,53,43,48,59E5
150 DATA 41,D2,54,52,41,4E,53,43,48,41,52,D1,43,4C,52,45,5F95
155 DATA 53,45,D4,43,4C,57,41,49,54,B1,43,4C,57,41,49,54,653A
160 DATA B2,43,41,50,53,D1,43,41,50,D3,44,49,53,43,D1,45,6BC4
165 DATA 52,52,CF,46,4F,52,4D,41,D4,55,53,45,52,B2,44,49,71FE
170 DATA 52,C5,48,45,41,C4,52,53,45,C3,57,53,45,C3,53,52,78AB
175 DATA 45,53,45,D4,53,49,4E,49,D4,57,49,4E,44,D1,50,45,7EFB
180 DATA 4E,D1,50,41,50,45,52,D1,4F,50,41,D1,4D,4F,44,D1,85C5
185 DATA 49,4E,4B,D1,42,4F,52,44,45,52,D1,53,50,49,4E,4B,8B8C
190 DATA D1,4C,49,4D,49,54,D1,52,45,D0,52,45,50,D1,43,48,9257
195 DATA 4F,CE,53,43,52,4F,4C,4C,B1,53,43,52,4F,4C,4C,B2,9875
200 DATA 53,50,52,49,54,45,B1,53,50,52,49,54,45,B2,46,49,9E15
205 DATA 4C,4C,C3,46,49,4C,4C,CF,4E,4F,49,D2,53,4F,43,54,A45/
210 DATA 45,D4,53,4F,43,54,45,54,B2,00,00,00,00,00,DD,7E,A94F
215 DATA 00,FE,01,20,06,3E,00,CD,5B,BD,C9,FE,02,20,06,3E,AEC4
220 DATA 04,CD,5B,BD,C9,FE,03,20,06,3E,05,CD,5B,BD,C9,FE,B68C
225 DATA 04,20,06,3E,06,CD,5B,BD,C9,FE,05,20,05,3E,07,CD,BBE2
230 DATA 5B,BD,C9,DD,5E,00,DD,56,01,DD,6E,02,DD,66,03,DD,C3A2
235 DATA 7E,04,47,2B,23,78,77,CD,B6,A1,FE,FF,28,F6,C9,DD,CC8D
240 DATA 4E,00,DD,46,01,DD,5E,02,DD,56,03,DD,6E,04,DD,66,D304
245 DATA 05,ED,B0,C9,DD,5E,00,16,00,DD,6E,02,DD,66,03,CD,DA20
250 DATA AB,BB,C9,DD,22,D1,A1,CD,AE,BB,DD,2A,D1,A1,EB,DD,E537
255 DATA 6E,00,DD,66,01,77,23,3E,00,77,DD,6E,02,DD,66,03,EACB
260 DATA 73,23,72,C9,DD,7E,00,FE,01,28,04,CD,03,BB,C9,CD,F243
265 DATA 00,BB,C9,DD,22,D1,A1,CD,06,BB,DD,2A,D1,A1,DD,66,FB82
270 DATA 01,DD,6E,00,77,C9,DD,22,D2,A1,3D,47,0E,00,DD,23,0212
275 DATA DD,23,CD,09,BB,32,D1,A1,79,FE,FF,28,11,3A,D1,A1,0AA2
280 DATA 4F,DD,7E,00,B9,28,06,DD,23,DD,23,10,F0,78,DD,2A,11B2
285 DATA D2,A1,DD,6E,00,DD,66,01,77,C9,DD,22,D1,A1,CD,21,1A53
290 DATA BB,44,4D,DD,2A,D1,A1,DD,6E,00,DD,66,01,78,77,23,21B9
295 DATA 3E,00,77,DD,6E,02,DD,66,03,79,77,C9,DD,7E,00,67,287C
300 DATA DD,7E,02,6F,CD,3A,BD,C9,DD,7E,02,DD,46,00,CD,39,305B
305 DATA BB,C9,DD,7E,02,CD,3C,BB,28,0E,DD,6E,00,DD,66,01,37C5
310 DATA 3E,01,77,3E,00,23,77,C9,3E,00,DD,6E,00,DD,66,01,3CE9
315 DATA 77,23,77,C9,3A,00,A7,DD,6E,00,DD,66,01,77,23,3E,430B
320 DATA 00,77,3A,01,A7,23,77,23,3E,00,77,3A,08,A7,23,77,4759
325 DATA 23,3E,00,77,3A,2C,A7,23,77,23,3E,00,77,3A,9F,A8,4C31
330 DATA 23,77,23,3E,00,77,3A,A0,A8,23,77,23,3E,00,77,3A,50D1
335 DATA A3,A8,23,77,23,3E,00,77,C9,DD,7E,00,32,D1,A1,21,5777
340 DATA BE,9E,CD,D4,BC,D0,3A,D1,A1,CD,1B,00,C9,DD,7E,00,60B8
345 DATA 32,01,A7,C9,B1,DD,7E,00,32,D1,A1,DD,7E,02,32,D2,683C
350 DATA A1,21,B5,A1,CD,D4,BC,D0,3A,D2,A1,5F,3A,D1,A1,CD,7206
```

```
355 DATA 1B,00,C9,CD,57,BB,DD,56,01,DD,5E,00,CD,9B,BC,CD,7A29
360 DATA 54,BB,C9,DD,22,D1,A1,DD,5E,00,DD,56,01,1A,47,13,8155
365 DATA 1A,6F,13,1A,67,DD,5E,02,DD,56,03,CD,77,BC,DD,2A,87EC
370 DATA D1,A1,DD,6E,04,DD,66,05,77,3E,00,23,77,23,73,23,8DFD
375 DATA 72,23,71,23,70,CD,7A,BC,C9,3E,07,32,E1,A1,21,66,94E2
380 DATA C6,22,DF,A1,DD,6E,00,DD,66,01,DD,4E,02,DD,56,04,9C3D
385 DATA DD,5E,06,DF,DF,A1,C9,3E,07,32,E1,A1,21,4E,C6,C3,A497
390 DATA 2B,9F,C9,CD,FF,BB,C9,CD,4E,BB,C9,CD,69,BB,24,2C,AE5A
395 DATA 14,1C,22,D1,A1,ED,53,D3,A1,DD,6E,00,DD,66,01,3A,B59B
400 DATA D4,A1,77,23,3E,00,77,DD,6E,02,DD,66,03,3A,D3,A1,BCA0
405 DATA 77,23,3E,00,77,DD,6E,04,DD,66,05,3A,D2,A1,77,3E,C2E8
410 DATA 00,23,77,DD,6E,06,DD,66,07,3A,D1,A1,77,3E,00,23,C8A1
415 DATA 77,C9,CD,93,BB,DD,6E,00,DD,66,01,77,23,3E,00,77,CFDA
420 DATA C9,CD,99,BB,DD,6E,00,DD,66,01,77,23,3E,00,77,C9,D76B
425 DATA CD,A2,BB,DD,6E,00,DD,66,01,FE,00,28,02,3E,01,77,DE02
430 DATA 3E,00,23,77,C9,CD,11,BC,DD,6E,00,DD,66,01,77,23,E466
435 DATA 3E,00,77,C9,DD,7E,04,DD,22,D1,A1,CD,35,BC,DD,2A,EC79
440 DATA D1,A1,DD,6E,00,DD,66,01,71,23,3E,00,77,DD,6E,02,F310
445 DATA DD,66,03,70,23,3E,00,77,C9,DD,22,D1,A1,CD,38,BC,FA9C
450 DATA DD,2A,D1,A1,DD,6E,00,DD,66,01,71,23,3E,00,77,DD,01CA
455 DATA 6E,02,DD,66,03,70,23,3E,00,77,C9,DD,22,D1,A1,CD,08CF
460 DATA 41,BC,DD,2A,D1,A1,EB,DD,6E,00,DD,66,01,73,3E,00,1070
465 DATA 23,77,DD,6E,02,DD,66,03,72,3E,00,23,77,C9,CD,17,1694
470 DATA BC,04,0C,C3,0E,A0,C9,DD,7E,00,FE,00,20,04,CD,57,1D3B
475 DATA BB,C9,CD,54,BB,C9,DD,7E,00,FE,01,20,09,CD,0B,BC,257B
480 DATA 23,23,CD,05,BC,C9,FE,02,20,09,CD,0B,BC,2B,2B,CD,2BF8
485 DATA 05,BC,C9,FE,03,20,0D,11,B0,07,CD,0B,BC,CD,C9,A1,3343
490 DATA CD,05,BC,C9,11,B0,07,CD,0B,BC,19,CD,05,BC,C9,DD,3B43
495 DATA 7E,00,DD,5E,02,DD,56,04,DD,6E,06,DD,66,08,DD,46,41F4
500 DATA 0A,25,2D,15,1D,CD,50,BC,C9,3E,00,32,DE,A1,DD,5E,484E
505 DATA 06,16,00,21,00,C0,19,2B,DD,46,04,11,50,00,05,78,4B94
510 DATA FE,00,28,03,19,10,FD,22,D4,A1,DD,5E,08,DD,56,09,51F9
515 DATA CD,FE,A0,2A,D4,A1,ED,53,D6,A1,11,50,00,19,22,D4,5A2A
520 DATA A1,ED,5B,D6,A1,DD,7E,00,3D,DD,77,00,FE,00,28,03,613F
525 DATA C3,DA,A0,C9,3E,08,06,00,DD,4E,02,ED,53,D8,A1,22,68F9
530 DATA DA,A1,32,DC,A1,3A,DE,A1,FE,01,28,01,EB,ED,B0,FE,728A
535 DATA 01,28,01,EB,2A,DA,A1,3A,DC,A1,11,00,08,19,22,DA,7829
540 DATA A1,2A,D8,A1,DD,4E,02,59,16,00,19,EB,2A,DA,A1,3D,7EEF
545 DATA FE,00,20,C4,C9,C9,3E,01,32,DE,A1,C3,B8,A0,C9,DD,8814
550 DATA 7E,00,DD,5E,02,DD,56,04,DD,6E,06,DD,66,08,25,2D,8DF4
555 DATA 1D,15,CD,44,BC,C9,DD,7E,00,DD,5E,02,DD,56,04,DD,9568
560 DATA 6E,06,DD,66,07,CD,47,BC,C9,3E,00,32,D1,A1,11,D1,9C83
565 DATA A1,CD,22,BD,C9,DD,6E,00,DD,66,02,25,2D,CD,1A,BC,A41E
570 DATA EB,DD,6E,04,DD,66,05,7B,77,7A,23,77,C9,DD,6E,00,ABBA
575 DATA DD,66,01,DD,5E,02,DD,56,03,CD,1D,BC,EB,DD,6E,04,B351
580 DATA DD,66,05,7B,77,7A,23,77,C9,C9,C9,83,7C,AA,7C,F2,BC11
585 DATA C1,A1,87,9F,D8,3C,C9,BA,20,F9,7D,93,20,F5,C9,B7,C5EE
590 DATA ED,52,37,E0,F6,FF,C9,00,00,00,00,00,00,00,00,00,CB02
595 DATA 00,00,00,00,00,00,00,00,00,00,C9,01,EE,A1,21,69,CDE5
600 DATA A2,C3,D1,BC,1A,A2,C3,6D,A2,C3,71,A2,C3,B9,A2,C3,D7EC
605 DATA B8,A2,C3,C7,A2,C3,D6,A2,C3,4A,A3,C3,87,A3,C3,C4,E2D1
610 DATA A4,C3,D7,A4,C3,DB,A4,C3,2E,A5,C3,4F,A5,C3,53,A5,ECFD
615 DATA 47,49,4E,49,D4,4F,52,49,D1,57,47,52,41,D1,47,50,F34C
620 DATA 45,4E,D1,47,50,41,50,45,52,D1,52,45,43,D4,54,52,F994
625 DATA 49,41,4E,C7,43,49,52,43,4C,C5,54,49,4D,C5,53,44,FFAB
630 DATA 52,45,53,45,D4,43,4F,4D,D0,44,45,43,4F,4D,D0,50,05E5
635 DATA 52,49,52,45,53,45,D4,50,52,49,42,55,53,D9,00,FC,0C2D
640 DATA A6,EE,A1,CD,BA,BB,C9,CD,CC,BB,44,4D,DD,6E,00,DD,167A
645 DATA 66,01,71,23,70,DD,6E,02,DD,66,03,73,23,72,C9,CD,1D16
650 DATA D5,BB,44,4D,DD,6E,02,DD,66,03,71,23,70,DD,6E,06,241F
655 DATA DD,66,07,73,23,72,CD,D8,BB,44,4D,DD,6E,00,DD,66,2BF0
660 DATA 01,71,23,70,DD,6E,04,DD,66,05,73,23,72,C9,CD,E1,330B
665 DATA BB,DD,6E,00,DD,66,01,77,23,3E,00,77,C9,CD,E7,BB,3ADC
670 DATA DD,6E,00,DD,66,01,77,3E,00,23,77,C9,DD,5E,06,DD,41A1
675 DATA 56,07,DD,6E,04,DD,66,05,CD,C0,BB,DD,5E,02,DD,56,494D
680 DATA 03,21,00,00,CD,F9,BB,11,00,00,DD,6E,00,DD,66,01,4E92
685 DATA CD,66,A5,CD,F9,BB,DD,6E,02,DD,66,03,CD,66,A5,EB,5841
690 DATA 21,00,00,CD,F9,BB,DD,5E,06,DD,56,07,DD,6E,04,DD,5F8A
695 DATA 66,05,CD,F6,BB,DD,7E,00,FE,01,28,01,C9,CD,C6,BB,6815
700 DATA 13,2B,13,2B,CD,C0,BB,21,D2,A5,11,05,00,CD,52,BD,6E63
705 DATA DD,5E,06,DD,56,07,DD,6E,04,DD,66,05,CD,C0,BB,C9,7686
710 DATA DD,5E,0A,DD,56,0B,DD,6E,08,DD,66,09,CD,C0,BB,DD,7ECD
715 DATA 5E,06,DD,56,07,DD,6E,04,DD,66,05,CD,F6,BB,DD,5E,86BB
720 DATA 02,DD,56,03,DD,6E,00,DD,66,01,CD,F6,BB,DD,5E,0A,8E45
725 DATA DD,56,0B,DD,6E,08,DD,66,09,CD,F6,BB,C9,00,CD,CC,9702
730 DATA BB,ED,53,72,A5,22,74,A5,DD,22,76,A5,DD,6E,00,DD,9F91
735 DATA 66,01,DD,5E,02,DD,56,03,CD,C9,BB,DD,6E,04,DD,66,A74E
740 DATA 05,11,78,A5,3E,00,CD,64,BD,FD,21,82,A5,06,26,21,AD3F
745 DATA 7D,A5,11,78,A5,CD,61,BD,11,D8,A5,E5,D5,FD,E5,C5,B769
750 DATA CD,85,BD,CD,6A,BD,C1,FD,E1,FD,75,00,FD,74,01,FD,C1EC
755 DATA 23,FD,23,21,7D,A5,11,78,A5,CD,61,BD,D1,21,05,00,C882
760 DATA EB,19,EB,E1,10,D5,ED,5B,82,A5,2A,84,A5,CD,C0,BB,D241
765 DATA 06,0B,DD,21,86,A5,C5,DD,5E,00,DD,56,01,DD,6E,02,D8FC
770 DATA DD,66,03,DD,23,DD,23,DD,23,DD,23,DD,E5,CD,F6,BB,E282
775 DATA DD,E1,C1,10,E1,06,0B,DD,21,AE,A5,C5,DD,6E,00,DD,EB41
780 DATA 66,01,CD,66,A5,EB,DD,6E,02,DD,66,03,DD,E5,CD,F6,F483
785 DATA BB,DD,E1,C1,DD,2B,DD,2B,DD,2B,DD,2B,10,DD,06,0B,FCDB
790 DATA DD,21,86,A5,DD,6E,00,DD,66,01,CD,66,A5,EB,DD,6E,05A1
795 DATA 02,DD,66,03,CD,66,A5,C5,DD,E5,CD,F6,BB,DD,E1,C1,1045
800 DATA DD,23,DD,23,DD,23,DD,23,10,DA,06,0C,DD,21,AE,A5,1792
805 DATA DD,5E,00,DD,56,01,DD,6E,02,DD,66,03,CD,66,A5,C5,1F31
810 DATA DD,E5,CD,F6,BB,DD,E1,C1,DD,2B,DD,2B,DD,2B,DD,2B,2A10
815 DATA 10,DE,21,00,00,11,00,00,CD,C0,BB,DD,2A,76,A5,DD,3077
820 DATA 7E,06,FE,01,20,09,21,D2,A5,11,05,00,CD,52,BD,ED,369A
825 DATA 5B,72,A5,2A,74,A5,CD,C9,BB,C9,DD,46,00,C5,06,32,3EB9
830 DATA 3E,FF,3D,FE,00,20,FB,10,F7,C1,10,F1,C9,CD,A7,BC,47DE
835 DATA C9,DD,56,01,DD,5E,00,DD,4E,02,DD,46,03,DD,6E,04,4EB8
840 DATA DD,66,05,D5,7E,23,BE,CC,11,A5,12,13,0B,CB,78,28,5551
845 DATA F3,AF,12,13,12,13,3C,12,E1,EB,ED,52,EB,DD,6E,06,5CD2
850 DATA DD,66,07,73,23,72,C9,D5,16,01,5F,14,28,13,23,08,61B5
855 DATA CB,78,20,04,7B,BE,28,F3,E3,73,23,73,23,7A,EB,E1,69C5
860 DATA C9,15,18,F4,DD,5E,00,DD,56,01,DD,6E,02,DD,66,03,70B1
865 DATA 7E,23,BE,CC,45,A5,C8,12,13,18,F5,23,46,05,C8,12,7708
870 DATA 13,10,FC,23,C9,CD,28,BD,C9,CD,2E,BD,38,04,3E,00,7DC0
875 DATA 18,02,3E,01,DD,6E,00,DD,66,01,77,C9,AF,95,6F,9C,8437
880 DATA 95,BC,67,37,C0,FE,01,C9,4F,41,44,20,37,30,30,30,8A69
885 DATA 48,0D,43,41,4C,4C,20,30,42,39,30,30,48,0D,43,41,8DDE
890 DATA 4C,4C,20,30,42,39,30,36,48,0D,30,00,48,0D,45,0D,90D6
895 DATA 4F,52,47,20,37,30,30,30,48,0D,4C,4F,41,44,20,37,9471
900 DATA 30,30,30,48,0D,43,41,4C,4C,20,30,42,39,30,30,48,97E5
905 DATA 0D,43,41,4C,4C,20,30,42,39,30,36,48,25,00,48,0D,9B04
910 DATA 45,0D,4F,52,47,20,37,30,30,30,48,0D,4C,00,00,00,9DC6
915 DATA 00,00,81,00,00,00,00,00,96,54,CF,7D,80,7C,A8,A8,A2C9
920 DATA 05,7E,3A,EA,46,77,80,0D,EE,83,04,7F,79,5E,83,6C,A974
925 DATA 80,35,15,EF,43,7F,42,D7,B3,5D,80,00,00,00,00,80,AF18
930 DATA 34,F3,04,35,80,34,F3,04,35,80,FB,C9,D7,1B,80,5A,B668
935 DATA 34,19,4B,80,00,00,00,00,80,42,D7,B3,5D,80,35,15,BAF3
940 DATA EF,43,7F,79,5E,83,6C,80,0D,EE,83,04,7F,3A,EA,46,C255
945 DATA 77,80,73,A8,A8,05,7E,97,54,CF,7D,80,00,00,00,00,CB49
950 DATA 00,00,00,00,00,81,C9,00,00,00,00,00,00,00,00,00,C993
```

QUICK 55

2ème partie



UN EXEMPLE : L'EDITEUR-COPIEUR DE SECTEURS

INTRODUCTION

Pour illustrer l'utilisation de ces RSX, je vous propose un exemple de programme, il s'agit d'un éditeur-copieur de secteurs disquette ; il a été entièrement réalisé en BASIC, à l'aide de QUICK55, en 104 lignes. EDITEUR est l'exemple d'une réalisation habituellement réservée aux programmeurs assembleur, maintenant accessible en BASIC.

• *UTILISATION*

EDITEUR nécessite la présence en permanence d'une disquette dans le lecteur ; il s'agit de votre disquette de travail, sur laquelle le menu principal, dans une fenêtre, vous donne des informations ; IL NE DOIT S'AGIR EN AUCUN CAS DE LA DISQUETTE COMPORTANT QUICK55, puisque des modifications peuvent avoir lieu, susceptibles d'effacer QUICK55. Le menu principal vous propose plusieurs options :

- Effacer un secteur : le secteur dont vous rentrez les coordonnées (n° de piste, n° de secteur) est effacé, et rempli avec l'octet de remplissage (&e5 au départ).

- Editer un secteur : le secteur dont vous entrez les coordonnées apparaît à l'écran (octets en hexadécimal, pour que les 512 octets soient visibles à l'écran). La frappe d'une touche vous fait revenir au menu principal.

- Copie de secteurs : vous entrez le numéro de la piste, les numéros de secteurs devant être copiés d'une disquette à l'autre. Le programme vous indique les disquettes (source ou but) à introduire, et attend après la frappe d'une touche.

- Copie intégrale : la face entière d'une disquette est recopiée sur une autre ; l'ordinateur vous indique les moments où vous devez changer de disquette. Remarque : la copie n'est possible que si la disquette est formatée de la même manière sur toute la face.

- Octet de remplissage : vous pouvez modifier l'octet avec lequel sont remplis les secteurs effacés.

L'ensemble de ces options est accessible en frappant le numéro lié à l'option.

• *STRUCTURE DU PROGRAMME*

- Chargement de QUICK55 ; initialisation

- Affichage du menu principal, après analyse de la disquette de travail (ùDISQ) ; le ùDIRE permet simplement d'avoir un premier contact avec la disquette sur laquelle vont s'effectuer les opérations.

- Attente d'une touche, envoi vers les différentes options.

- OPTION EFFACER : le ùFILLC et le ùRECT permettent d'obtenir un rectangle en surimpression (procédé utilisé également après). Le secteur virtuel est placé de 16384 à 16895. Il est rempli avec l'octet de remplissage, puis le secteur virtuel est sauvegardé.

- OPTION EDITION : le secteur virtuel est placé au même endroit que précédemment, puis il est affiché.

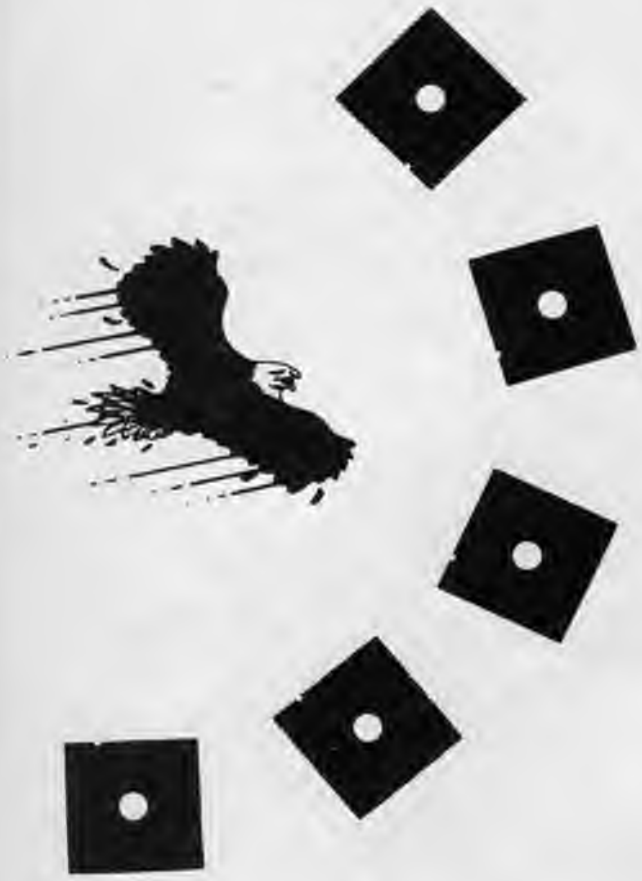
- OPTION MODIFICATION : chargement du secteur à partir de 16384, modifications par l'utilisateur, puis sauvegarde.

- OPTION COPIE SECTEURS : la piste ayant été choisie, les secteurs concernés sont chargés à partir de 16384, puis sauvegardés sur l'autre disquette.

- OPTION COPIE INTEGRALE : 2 fois de suite, 15 pistes sont chargés dans les blocs de mémoire parallèle et sauvegardées sur la disquette destination. Pour les 10 dernières pistes, le principe est le même.

- OPTION OCTET DE REMPLISSAGE : l'octet est modifié.

Jean Louis Benard



EDITEUR

```
10 MODE 2:MEMORY 16383:LOAD "QUICK55.BIN":CALL &9BAA:CA >KA
LL &A1E5
20 DIM DIS%(6)                                          >MH
30  :DIRE,16384:!DISCQ,@DIS%(@)                         >AT
40 REM PRESENTATION                                     >QK
50 MODE 2                                               >BJ
60 INK @,23:BORDER 23:INK 1,@                           >WJ
70 GOSUB 960                                            >CC
80 LOCATE 14,5 :PRINT "E D I T E U R / C O P I E U R  D >FB
 E  S E C T E U R S":!RECT,100,343,415,28
90 LOCATE 9,11:PRINT "1) EFFACER UN SECTEUR"            >PD
100 LOCATE 9,13:PRINT "2) EDITER UN SECTEUR"            >NE
110 LOCATE 9,15:PRINT "3) MODIFIER UN SECTEUR"          >RQ
120 LOCATE 9,17:PRINT "4) COPIER SECTEURS"              >ML
130 LOCATE 9,19:PRINT "5) COPIE INTEGRALE"              >LJ
140 LOCATE 9,21:PRINT "6) OCTET REMPLISSAGE"            >PN
150 !RECT,290,300,327,280                               >TT
160 LOCATE 48,8:PRINT "ETAT DE LA DISQUETTE"            >PB
170 LOCATE 40,12:PRINT "NO DE PREMIER SECTEUR: ";USING  >TQ
"###";DIS%(4)
180 LOCATE 40,15:PRINT "NBRE DE SECTEURS / PISTE: ";USI >WA
NG "##";DIS%(5)
190 LOCATE 40,18:PRINT "OCTET DE REMPLISSAGE: ";USING " >RJ
###";DIS%(6)
200 LOCATE 40,21:PRINT "NO DE PISTE,SECTEUR EN COURS:"; >HB
P;S
210 C%=@:!CLWAIT2,49,50,51,52,53,54,@C%                 >GX
220 ON C%+1 GOTO 210,230,320,500,630,750,970            >HZ
230 REM EFFACER                                         >FH
240 !FILLC,25,@,60,16,@                                 >RV
250 !RECT,200,256,280,112                               >TL
260 LOCATE 30,11:PRINT "EFFACER SECTEUR"                >KB
270 LOCATE 27,13:INPUT "NUMERO DE PISTE(@ -> 39) ";NP   >YK
280 LOCATE 27,15:PRINT "NUMERO SECTEUR(";DIS%(4);"->";D >TB
IS%(4)+DIS%(5)-1;")";:INPUT NS
290 IF NP<@ OR NP>39 OR NS<DIS%(4) OR NS>DIS%(4)+DIS%(5 >KZ
)-1 THEN 330
300 FOR I=16384 TO 16895:POKE I,DIS%(6):NEXT I:!WSEC,@, >PN
NP,NS,16384
310 P=NP:S=NS:GOTO 40                                   >QX
320 REM EDITION                                         >LK
330 !FILLC,25,9,60,16,@                                 >RV
340 !RECT,200,256,280,112                               >TL
350 LOCATE 30,11:PRINT "E D I T I O N"                  >EG
360 LOCATE 27,13:INPUT "NUMERO DE PISTE(@ -> 39) ";NP   >YK
370 LOCATE 27,15:PRINT "NUMERO SECTEUR(";DIS%(4);"->";D >TB
IS%(4)+DIS%(5)-1;")";:INPUT NS
380 IF NP<@ OR NP>39 OR NS<DIS%(4) OR NS>DIS%(4)+DIS%(5 >K2
)-1 THEN 330
390 !RSEC,@,NP,NS,16384                                 >TN
400 K=3:MODE 2:GOSUB 960:LOCATE 24,1:PRINT "PISTE : ";N >KT
P;"      ";"SECTEUR : ";NS
410 FOR I=16384 TO 16896 STEP 25                        >XJ
420 LOCATE 4,K                                          >TB
430 FOR J=@ TO 24                                       >CE
440 IF I+J>16895 THEN 490                               >RF
450 A$=HEX$(PEEK(I+J)):IF LEN(A$)=1 THEN A$="@"+A$:     >TX
460 PRINT A$;" ";:NEXT J                                >RR
470 K=K+1                                               >DK
480 NEXT I                                              >PA
490 P=NP:S=NS:GOSUB 1040:GOTO 40                        >BD
500 REM MODIFICATION                                    >QR
510 MODE 2:GOSUB 960                                    >NF
520 LOCATE 30,3:PRINT "MODIFICATION SECTEURS"           >RA
530 LOCATE 4,8  :INPUT "NUMERO DE PISTE(@ -> 39) ";NP   >XD
540 LOCATE 4,10 :PRINT "NUMERO SECTEUR(";DIS%(4);"->";D >RD
IS%(4)+DIS%(5)-1;")";:INPUT NS
550 IF NP<@ OR NP>39 OR NS<DIS%(4) OR NS>DIS%(4)+DIS%(5 >KX
)-1 THEN 500
560 !RSEC,@,NP,NS,16384                                 >TN
570 LOCATE 4,12:INPUT "COMBIEN D'OCTETS A MODIFIER";NO  >CB
580 FOR I=1 TO NO:LOCATE 4,14:INPUT "NUMERO D'OCTET(1 - >PG
> 512)";NDO
590 LOCATE 4,16:PRINT "ANCIENNE VALEUR(DECIMALE): ";:PR >DA
INT PEEK(16383+NDO)
600 LOCATE 4,18:INPUT  "NOUVELLE VALEUR";V:POKE 16383+I >DW
,V
610 NEXT I                                              >NF
620 !WSEC,@,NP,NS,16384:S=NS:P=NP:GOTO 40               >LE
630 MODE 2:GOSUB 960                                    >MJ
640 LOCATE 35,3:PRINT "COPIE SECTEURS"                  >HW
650 LOCATE 4,8  :INPUT "NUMERO DE PISTE(@ -> 39) ";NP   >XG
660 LOCATE 4,10 :PRINT "NUMERO DE SECTEUR DE DEPART(";D >PK
IS%(4);"->";DIS%(4)+DIS%(5)-1;")";:INPUT NS1
```

```
670 LOCATE 4,12 :PRINT "NUMERO DE SECTEUR D'ARRIVEE(";D >PG
ISX(4);"-)";DISX(4)+DISX(5)-1;")";:INPUT NS2
680 IF NP<0 OR NP>39 OR NS1<DISX(4) OR NS1>DISX(4)+DISX >QN
(5)-1 OR NS2<DISX(4) OR NS2>DISX(4)+DISX(5)-1 OR NS2<NS
1 THEN 630
690 LOCATE 4,14: PRINT "INSEREZ DISQUETTE SOURCE":GOSUB >JA
 1040
700 ADR=16384:FOR I=NS1 TO NS2:|RSEC,0,NP,I,ADR:ADR=ADR >RU
+513:NEXT I
710 LOCATE 4,16:  PRINT "INSEREZ DISQUETTE BUT":GOSUB 1 >EG
040
720 ADR=16384:FOR I=NS1 TO NS2:|WSEC,0,NP,I,ADR:ADR=ADR >RB
+513:NEXT I
730 P=NP:S=NS1                                          >EG
740 GOTO 40                                             >TB
750 REM COPIE INTEGRALE                                 >TA
760 ADR=16384:NP=0:FOR K=1 TO 3                         >YX
770 |FILLC,25,9,60,16,0                                 >RD
780 |RECT,200,256,280,112                               >TV
790 LOCATE 35,11:PRINT "COPIE INTEGRALE"                >KV
800 LOCATE 27,13:PRINT "INSEREZ LA SOURCE":GOSUB 1040   >ZR
810 FOR L=1 TO 5                                        >WD
820 IF NP=39 THEN GOSUB 940:GOTO 860                    >CV
830 |CONNECT,L                                          >LZ
840 FOR M=0 TO 2:FOR N=0 TO 8:|RSEC,0,NP,DISX(4)+N,ADR: >XB
ADR=ADR+513:NEXT N:NP=NP+1:NEXT M
850 ADR=16384:NEXT L                                    >PN
860 LOCATE 27,15:PRINT "INSEREZ LE BUT":GOSUB 1040:ADR= >JB
16384:IF NP<>39 THEN NP=NP-15 ELSE NP=NP-9
870 FOR L=1 TO 5                                        >WK
880 IF NP=39 THEN GOSUB 950:GOTO 930                    >CA
890 |CONNECT,L                                          >LF
900 FOR M=0 TO 2:FOR N=0 TO 8:|WSEC,0,NP,DISX(4)+N,ADR: >XD
ADR=ADR+513:NEXT N:NP=NP+1:NEXT M
910 ADR=16384:NEXT L                                    >PK
920 ADR=16384:NEXT K                                    >PK
930 GOTO 40                                             >TC
940 ADR=32770:FOR N=0 TO 8:|RSEC,0,39,DISX(4)+N,ADR:ADR >GN
=ADR+513:NEXT N:RETURN
950 ADR=32770:FOR N=0 TO 8:|WSEC,0,39,DISX(4)+N,ADR:ADR >GV
=ADR+513:NEXT N:RETURN
960 |RECT,1,399,638,398:|RECT,10,389,618,378:RETURN     >YC
970 REM OCTET REMPLISSAGE                               >VG
980 |FILLC,25,9,60,16,0                                 >RG
990 |RECT,200,256,280,112                               >TY
1000 LOCATE 30,11:PRINT "OCTET REMPLISSAGE"             >NZ
1010 LOCATE 27,13:INPUT "OCTET REMPLISSAGE";DISX(6)     >XA
1020 IF DISX(6)<0 OR DISX(6)>255 THEN 970               >FV
1030 GOTO 40                                            >YC
1040 I%=0:|CLWAIT1,0X%:RETURN                           >AR
```