RÉALISER UN MICRO-ORDINATEUR "HAUT DE GAMME"

Vegas 6809

VIII. LES INSTRUCTIONS DE GESTION DE FICHIERS.

Nous l'avons dit le mois dernier : le XBasic est un langage complet et performant, adapté aux calculs scientifiques (de par l'amplitude des nombres qu'il gère) et aux applications de gestion. Ces dernières sont particulièrement aisées à élaborer grâce au puissant jeu d'instructions disponible.

En outre, l'utilisateur a la possibilité d'exécuter certaines commandes du FLEX depuis son programme même, ce qui accroît considérablement cette puissance (tout en étant pourtant un facteur de risque pour les programmeurs maladroits).

L'exploitation d'une commande du Flex est autorisée par l'utilisation de l'instruction « EXEC » du XBasic. Ainsi, par exemple :
10 EXEC, « TTYSET, WD=0 »
exécute la commande « TTYSET » de la même manière que si elle avait été appelée par l'utilisateur se trouvant sous Flex. En l'occurrence, elle permet de supprimer le passage à la ligne automatique après la fin du premier mot dépassant la 65e colonne.

Il est primordial de veiller à ce que la commande appelée n'interfère pas avec le XBasic. Cela signifie qu'elle ne doit pas altérer les zones mémoires comprises entre les adresses hexadécimales 0000 et BFFF. De ce fait, toutes les commandes ne pourront pas être utilisées (par exemple COPY, FORMAT ne doivent pas être exécutées).

Les commandes du Flex peuvent en outre être appelées en mode interprétation (c'est-à-dire en dehors de l'exécution d'un programme ; après, on commande RUN). Pour ce faire, le caractère « + » doit être employé au lieu de EXEC. Ainsi,

+TTYSET, WD = 0

aura le même rôle que l'instruction du numéro 10 citée ci-dessus.

Les utilitaires de XBasic

Pour pallier l'impossibilité d'exploiter les principales commandes Flex de gestion de disquettes, XBasic propose directement un jeu d'instructions permettant de résoudre la majorité des cas présentés à l'utilisateur.

KILL est utilisée pour détruire un fichier sur une disquette. L'opérande de cette instruction est le nom de fichier qui peut être complet (incluant le numéro de disquette et la valeur de l'extension) ou seulement composé du nom lui-même (auquel cas, l'extension est supposée être « BAS » et le numéro celui de la disquette donnée par défaut). Détruire le fichier Vegas dont l'extension est TXT se fera, par exemple, par :
10 A$ = « VEGAS . TXT »
20 KILL A$

RENAME permet de rebaptiser un fichier. Les deux opérandes de cette instruction sont respectivement l'ancien puis le nouveau nom, séparés par une virgule. Il faut observer que si la valeur de l'extension n'est pas fournie, « BAS » sera assumé. Ainsi :
10 RENAME «VEGAS», «SOS»
changera, s'il existe, le nom du fichier « VEGAS.BAS » en « SOS.BAS ».

CHAIN permet de lancer un programme XBasic depuis un autre. Cette fonction permet de pallier une éventuelle limitation de mémoire par un enchaînement des divers modules fonctionnels.

Il est à remarquer toutefois qu'aucun paramètre ne peut être convoyé ainsi entre deux programmes : cette instruction réinitialise en effet la totalité de la mémoire, ferme tous les fichiers éventuellement ouverts, puis charge le programme appelé. Il est recommandé, si les données doivent être transmises, de les écrire sur un fichier de travail.

Enfin, il est possible de préciser comme second opérande de l'instruction CHAIN le numéro de la ligne à laquelle le programme appelé devra démarrer.

La gestion des fichiers

Trois types de gestion sont disponibles avec XBasic : **l'accès séquentiel** (classique), **l'accès di-**

Les tableaux virtuels, sorte de système de gestion de fichiers à accès direct, représentent une des performances les plus intéressantes de XBasic.

rect et les **tableaux virtuels,** un procédé intéressant de gestion de mémoire virtuelle.

L'accès séquentiel est le mode le plus simple. Les données sont inscrites dans les fichiers les unes après les autres. Ce procédé, le plus ancien, présente certains inconvénients : l'accès à la donnée « n » est subordonné à la lecture préalable des « n–1 » données précédentes, les modifications « directes » sont impossibles et l'agrandissement de leur taille nécessite l'emploi de commandes particulières.

Quatre instructions permettent leur gestion. Ce sont « **OPEN** », « **CLOSE** », « **PRINT#** » et « **INPUT#** ».

OPEN a pour objet d'initialiser les entrées/sorties en « ouvrant » le fichier et en lui affectant un numéro logique qui sera utilisé ultérieurement par les instructions de lecture/écriture. La syntaxe de cette instruction est :

OPEN NEW / OLD « **Nom** » **AS** n

(avec $1 \leqslant n \leqslant 12$)

Sachant qu'un numéro logique ne peut être associé qu'à un seul fichier, il apparaît donc que XBasic peut traiter au maximum douze fichiers simultanément.

Le rôle des paramètres NEW et OLD est de préciser l'utilisation qui va être faite du fichier. NEW précise que celui-ci est nouveau, et, de ce fait, si un nom identique à celui fourni par OPEN existe sur la disquette, il sera alors **détruit.** Il faut remarquer qu'un tel fichier ne peut pas être lu : puisqu'il est en cours de création, il ne contient rien et ne peut donc accepter que des ordres d'écriture. Le paramètre OLD indique que le fichier existe déjà et qu'il sera accédé en lecture.

L'absence de paramètre NEW ou OLD permet de créer le fichier s'il n'existe pas ou de l'ouvrir simplement s'il existe.

CLOSE indique que les entrées/sorties associées à un fichier sont terminées. En cas de fichier en cours d'écriture, le dernier bloc est ajouté sur la disquette. L'association « numéro logique - nom du fichier » est détruite, et l'utilisation de ce numéro est proscrite jusqu'à une nouvelle ouverture.

PRINT# est l'instruction d'écriture. Le « # » est toujours suivi du numéro logique du fichier ouvert préalablement sur lequel l'écriture doit se faire.

Les opérandes qui suivent indiquent ce qui doit être écrit (nombres, chaînes, etc.)

INPUT# est l'instruction de lecture. Le « # » est immédiatement suivi du numéro logique du fichier sur lequel s'effectue la lecture. Les valeurs lues sont affectées aux variables dont les noms sont fournis en opérande de l'instruction.

Le programme suivant présente les quatre instructions :

```
10 OPEN NEW « VEGAS »
   AS 7
20 PRINT#7, « ESSAI DE
   L'ORDINATEUR »
30 PRINT#7, « .VEGAS
   6809 »
40 CLOSE 3
50 OPEN OLD « VEGAS »
   AS 3
50 INPUT#7, C$
70 INPUT#7, D$
80 CLOSE 3
90 END
```

A la fin de ce programme, les variables C$ et D$ auront le même contenu que A$ et B$ et que le fichier « VEGAS . DAT » (l'extension DAT étant prise par défaut pour les fichiers ouverts depuis le XBasic).

Les tableaux « virtuels »

La notion de « **tableaux virtuels** » recouvre une catégorie spéciale de fichiers. En fait, ils permettent d'accéder **directement** à une donnée sans s'encombrer de la gestion de fichiers à accès direct classiques.

En fait, ils sont réellement utilisés comme des tableaux, mais leur taille n'est plus limitée par la mémoire de l'ordinateur, ce qui peut être très pratique pour les tableaux de calculs ou certaines applications scientifiques. En outre, chaque poste peut être lu et/ou modifié à discrétion par l'utilisateur. De plus, leur « vie » s'étend au-delà de leur utilisation (du fait du stockage sur un périphérique magnétique tels la disquette ou le disque dur).

Leur manipulation s'effectue par les trois instructions OPEN, CLOSE et DIM# ainsi qu'à l'aide des instructions habituelles d'affectation de valeurs à un tableau.

OPEN agit de la même manière que pour un fichier séquentiel, c'est-à-dire qu'elle affecte un numéro logique à un nom de fichier qui sera le tableau virtuel.

CLOSE a encore le rôle de fermeture d'un fichier, donc de libération d'un numéro logique.

DIM# est l'instruction caractérisant les tableaux virtuels. De la même manière que les tableaux normaux, cette instruction définit leur taille. Il faut se souvenir que l'encombrement d'un tableau virtuel est égal au produit du nombre de postes de ce dernier par la taille de chaque poste.

Trois types de tableaux virtuels peuvent être utilisés :

```
10 OPEN « nom », 3
20 DIM# 3, A (100,50)
```

définit un tableau de valeurs **réelles,** dont chaque poste occupe 8 octets ;

```
10 OPEN « nom », 7
20 DIM# 7, B% (30)
```

définit un tableau de valeurs **entières,** dont chaque poste occupe 2 octets ;

```
10 OPEN « nom », 10
20 DIM# 10, A$ (300) = 30
```

définit un tableau de **chaînes de caractères,** dont chaque poste occupe 30 octets. Si la longueur de chaque chaîne n'est pas précisée, la longueur par défaut sera de 18. La longueur maximum autorisée pour une chaîne de caractères dans un tableau virtuel est de 252, soit celle d'un secteur du disque. Lors de la création de ces tableaux, il faudra se souvenir qu'un secteur ne peut conte-

nir qu'un multiple de postes et que, si la longueur de ceux-ci n'est pas calculée correctement, de la place sur disque peut être perdue. Ainsi, par exemple :
DIM # 7, D$ (10) = 127
« consommera » dix secteurs (127 × 2 > 252, donc il n'y aura qu'un poste par secteur, ce qui correspond à une perte de 125 octets par poste utilisé), tandis que :
DIM #7, D$ (10) = 126
n'en consommera que cinq, sans aucune perte d'octet. L'utilisation d'un tableau a lieu par l'emploi du nom fourni lors de l'instruction DIM # :
10 A (10,1) = 3 . 14
ou encore :
10 A (153) = « VEGAS, REALISATION MICRO-SYSTEMES »

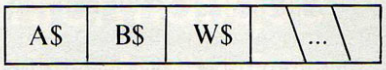
Enfin, l'utilisation d'un poste comme **valeur** n'est possible que si une valeur lui a été affectée. Ainsi :
100 PRINT A (100,50)
est interdit après l'ouverture en création, aucune valeur n'ayant encore été affectée. Par contre :
100 A (100,50) = 0
a pour effet de créer le dernier poste du tableau... et par conséquent tous les précédents.

« **L'accès direct** » constitue le dernier mode d'accès proposé avec XBasic. D'un emploi plus compliqué que les deux précédents, il s'avère très utile pour les utilisations de bases de données. Neuf instructions sont utilisées pour leur exploitation, excepté OPEN et CLOSE dont le rôle est le même que dans les deux autres organisations.

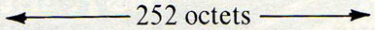
L'accès direct ne traite pas des articles ou enregistrements, mais manipule des secteurs (donc des entités de 252 octets) que l'utilisateur devra veiller à structurer.

FIELD# a pour rôle d'assurer cette structuration. Son utilisation permet de décrire un « masque » pour chaque secteur du fichier.

Ainsi :
FIELD# 1, 20 AS A$, 10 AS B$, 6 AS W$ définit le secteur suivant, associé au numéro logique 1, ouvert préalablement :



20 octets 10 octets 6 octets inutilisé

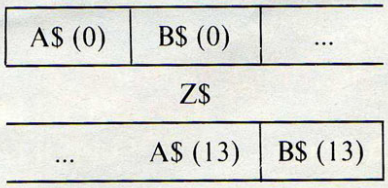


Bien entendu, cette instruction doit être exécutée avant toute utilisation d'un fichier ouvert.

La constitution de sous-enregistrements accessibles par l'intermédiaire de tableaux est aussi offerte avec l'instruction FIELD#. Ainsi :

```
 10 OPEN « FILE » AS 1
100 DIM A$ (13), B$ (13)
110 FOR I% = 0 TO 13
120 FIELD#4, I%*18 AS Z$,
    10 AS A$ (I%), 8 AS B$
    (I%)
130 NEXT I%
```

créera pour le fichier logique 1, associé au nom « FILE-DAT », une structure suivante pour chaque secteur :



L'utilisation des variables affectées par FIELD# doit se faire par l'intermédiaire des instructions spécifiques décrites ci-dessous, sans quoi la relation entre leur nom et leur position dans le masque de l'enregistrement sera détruite...

PUT# permet d'écrire un enregistrement complet (un secteur) sur le fichier. Sa syntaxe est :
PUT# n [, RECORD n°]
où n est le numéro du secteur où devra être écrit l'enregistrement. Ce numéro est en fait le rang du secteur visé dans le fichier (premier secteur de numéro 1, etc.). Si aucun numéro n'est fourni, une écriture **séquentielle** sera exécutée.

GET# a pour fonction de lire le secteur dont le numéro est précisé (si ce secteur n'existe pas, un message d'erreur est émis). La syntaxe de l'instruction est la même que celle de PUT#.

L'affectation de valeur est possible pour chaque partie de l'enregistrement, en utilisant les noms fournis lors de l'instruction FIELD#.

LSET est une instruction équivalente à LET, mais effectue une affectation avec justification à gauche (LSET AB$ = W$).

RSET, à l'opposé, effectue une affectation avec justification à droite.

La manipulation des valeurs numériques n'est toutefois pas possible avec seulement ces deux instructions. Des fonctions de

Comment réaliser Vegas 6809 ?

L'ensemble des éléments nécessaires à la construction de Vegas :
– kit de base (carte « mère » avec ses composants, lecteur de disquettes, clavier Qwerty, système d'exploitation Flex et XBasic) ;
– le circuit imprimé ;
– les composants ;
– le (ou les) lecteur(s) de disquettes ;
– le clavier...
est disponible par correspondance chez son concepteur, **Microkit,** B.P. 46, 91302 Massy Cedex. Tél. : (1) 681.88.37.

Vous pouvez également voir Vegas chez :
– **SOS Computer,** 78, rue de Dunkerque, 75009 paris. Tél. : (1) 281.03.73.
– **Lens Buro,** 73, boulevard Basly, 43200 Lens. Tél. : (21) 28.39.43.

Vegas est une marque déposée 3D International, 2, rue de l'Armée-Patton, 91640 Briis-sous-Forges. Tél. : (1) 594.61.36.

conversion ont donc été implémentées :

CVTF$** et **CVT$% ont comme rôle d'assurer l'affectation respectivement d'un nombre flottant ou d'un nombre entier à une portion d'un secteur :

A$ = CVTF$ (Y)
ou B$ = CVT%$(X%)

CVT$F et **CVT%$** assurent la fonction inverse, à savoir la transformation d'une portion de secteur en un nombre flottant (ou respectivement en un nombre entier).

W = CVT$F (B$) ou X% = CVT$% (W$)

L'exploitation des fichiers à accès direct étant un peu plus délicate que celle des autres organisations, nous vous proposons ici un exemple, relativement simple, montrant toutefois une utilisation classique. Le but de ce programme est d'imprimer (ou plutôt d'afficher à l'écran) le nom et le numéro de téléphone d'un individu dont on connaît le numéro. La structure du fichier est la suivante :

20 caractères pour le nom
69 caractères pour l'adresse
15 caractères pour le numéro

Le programme autorise, de plus, des modifications du fichier :

```
10 OPEN « EMPLOYE » AS 1
20 FIELD#1, 20 AS N$, 69 AS D$, 15 AS P$
30 INPUT « NUMERO DE L'EMPLOYE », E%
40 GET#1, RECORD E%
50 PRINT N$, P$
60 INPUT « CHANGEMENT DE NUMERO », R$
70 IF R$ < > « OUI » THEN 110
80 INPUT « NOUVEAU NUMERO », A$
90 LSET P$ = A$
100 PUT# 1, RECORD E%
110 CLOSE 1
120 END
```

Conclusion

La présentation du XBasic se termine ici. Nous l'avons vu, c'est un interpréteur puissant et déjà relativement rapide. Ses performances permettent à tout utilisateur d'élaborer la majorité des programmes dont il peut avoir besoin, depuis des logiciels de traitement de texte simplifiés jusqu'à des logiciels de gestion de fichiers élaborés. Bien sûr, ces outils, s'ils suffisent souvent aux utilisateurs, sont laborieux à écrire, et de nombreux logiciels utilitaires, tels des tris, des bases de données ou des éditeurs de texte, développés sous Flex, sont disponibles dans le commerce. ■

N. NUTIN, D. HABERT