LA GESTION DES FICHES SOUS SEDORIC

par Yann Legrand et André Chéramy
- Basé sur un article de Ray MacLaughlin -
(suite et fin)

Passons maintenant aux explications des commandes de gestions des fiches sous Sédoric

OPEN R,Nom_de_fichier,Numéro_Logique(,Longueur_de_chaque_Fiche,Nombre_de_fiches_à_Réserver) soit en abrégé: **OPEN R,NF,NL(,LF,NR)** pour ouvrir un fichier à accès direct (**R**andom) de nom **NF**, de n° logique **NL**, comportant un **N**ombre de fiches à **R**éserver **NR**, la **L**ongueur de chaque Fiche étant de **LF** caractères. Attention **LF** et **NR** ne sont à préciser que la 1ère fois, lors de la création du ficher à accès direct. Mais ils sont alors obligatoires. Si nécessaire, l'extension au delà de ce nombre de fiches est automatique dans la limite de la place disponible sur la disquette. La place occupée sur la disquette par le fichier **NF** est égale (environ) au nombre d'octets de data, soit **NR** x **LF** / 256 secteurs.

Le "Pseudo Tableau" **FI** est initialisé si cela n'avait pas encore été le cas. Si le **NL** est déjà attribué, on obtient une "FILE ALREADY OPEN ERROR". Sinon, le "Channel Buffer" correspondant au **NL** est généré, comme expliqué auparavant. Si le fichier n'existe pas encore, il est créé avec le **N**ombre de fiches **R**éservées **NR** spécifié, ayant chacune la **L**ongueur de Fiche indiquée, de **LF** caractères. Si par contre le fichier existe sur la disquette, on vérifie qu'il est du bon type (**R**, sinon on a une "FILE TYPE MISMATCH ERROR"), puis si c'est le cas on l'ouvre avec **NL** (la longueur des fiches est déjà indiquée dans le fichier).

Dans les deux cas, les fiches qui n'ont pas été manipulées par la commande PUT, contiennent malgré tout des résidus d'informations qui ont toutes les chances de ne pas avoir la structure définie par la commande FIELD. Bien qu'il soit possible de lire ces fiches par TAKE, il n'est donc pas facile d'y accéder.

TAKE NL, n°_de_fiche charge en mémoire la fiche indiquée qui doit bien sûr exister, sinon “BAD RECORD NUMBER ERROR”. Le fichier doit avoir été ouvert, sinon “FILE NOT OPEN ERROR”. Le fichier ouvert avec OPEN et celui indiqué par TAKE doivent être du même type, sinon “FILE TYPE MISMATCH ERROR”.

Si tout est correct, 2 secteurs de la disquette sont lus dans le “General Buffer” (le 1er secteur contient le début de la fiche choisie). La fiche proprement dite est alors copiée du “General Buffer” dans le “Chanel’s own Data Buffer”. En fait #100 octets au total sont copiés, à partir du début de la fiche, c’est à dire généralement la fiche plus quelques octets, appartenant à la fiche suivante. Cette méthode est plus simple et plus compacte (en taille de programme) que d’utiliser un compteur basé sur la longueur réelle de la fiche.

PUT NL,n de fiche, copie la fiche présente en mémoire sur la disquette, dans le fichier indiqué par **NL**, au n° de fiche indiqué. Si la fiche n'existe pas, elle sera créée, s'il y a encore de la place sur la disquette, ainsi que toutes les fiches de n° inférieur qui n'existent pas encore. Conclusion, pour épargner de la place, il vaut mieux créer des fiches de n° consécutifs.

Le fichier doit avoir été ouvert, sinon "FILE NOT OPEN ERROR". Le fichier ouvert avec OPEN et celui indiqué (**NL**) par PUT doivent être du même type, sinon "FILE TYPE MISMATCH ERROR". Par sécurité, l'écriture sur la disquette se fait immédiatement après chaque PUT.

Si tout se déroule sans problème, les data sont lus de 2 secteurs consécutifs de la disquette (le 1er secteur contient le début de la fiche choisie) dans le "Buffer Général". La fiche est alors recopiée, à sa place exacte, du "Channel's own Data Buffer" dans le "Buffer Général". Contrairement à ce qui se passe pour la commande TAKE, la longueur réelle de la fiche sert ici pour le décompte du nombre d'octets recopiés, sous peine d'écraser le début de la fiche suivante. Enfin, les 2 secteurs sont réécrits, à leur place, sur la disquette.

FIELD NL,nom_de_champ TO type (,nom_de_champ TO type ,...), cette commande est utilisée afin de définir le ou les différents champs d'une fiche modèle. Pour le fichier indiqué par **NL**, elle associe donc à chaque nom de champ indiqué un type qui peut être alphanumérique (type $), entier (type %), réel (type !) ou octet (type O).

Pour définir les différents champs d'une fiche modèle, il faut soit indiquer ces différents champs dans la même commande FIELD, soit utiliser plusieurs commandes FIELD, mais alors chacune de ces commandes (sauf la dernière) devra se terminer par une virgule.

Dans le cas d'un champ alphanumérique, le signe $ doit être suivit de la longueur maximale, qui ne peut évidement excéder la place restant disponible dans la fiche. La longueur d'un champ réel est de 5 octets,

celle d'un champ entier 2 octets, celle d'un champ octet 1 octet. De plus, il faut rajouter 2 octets par champ pour les informations de gestion interne. Attention donc à la place disponible dans une fiche! Le fichier doit évidemment être ouvert, sinon "FILE NOT OPEN ERROR" et être du bon type, sinon "FILE TYPE MISMATCH ERROR".

Les informations concernant les champs d'une fiche modèle sont gardées dans le "Field Buffer" et la place totale occupée par tous ces champs est calculée lorsque de nouveaux champs sont définis pour la fiche modèle du fichier. Si la commande FIELD s'achève par une virgule, la prochaine commande FIELD, se référant au même **NL**, révisera la place totale occupée par les champs. Si ce nombre devient plus grand que celui spécifié pour ce **NL**, on a une "FIELD OVERFLOW ERROR". En absence de virgule terminale, la commande FIELD suivante, pour le même **NL**, remet ce compteur à zéro (redéfinition de la fiche modèle).

Non ou mal documenté:

1) Attention le "TO" de la commande FIELD, comme celui des autres commandes Sédoric doit être en majuscule, sinon il n'est pas reconnu! De même, on ne peut pas taper le "O" de type octet en minuscule. Il y a un peu de flottement dans l'usage des minuscules pour les commandes Sédoric!

2) Il est possible d'utiliser n'importe quoi (sauf les deux points) comme délimiteur entre les définitions successives des noms de champ et pas seulement une virgule! (bogue située en FC48 et qui pourrait être réparée par un JSR D22C).

3) Il est possible d'indexer un nom de champ, comme s'il s'agissait d'un élément de tableau, NOM(2) par exemple, sans que cela crée pour autant les autres éléments du tableau: NOM(0) et NOM(1). En effet, il s'agit seulement d'un système de notation qui permet de compléter le nom du champ (5 caractères maximum) et non de DIMensionner un vrai tableau. Dans notre exemple, FIELD 1, NOM(2) TO $8 ne défini que le seul champ NOM(2) comme étant un champ alphanumérique de 8 caractères de long. Pour définir aussi NOM(0) et NOM(1), il faudra taper:

FIELD 1, NOM(0) TO $8, NOM(1) TO $8, NOM(2) TO $8 ou plus rapidement:

FOR I=0 TO 2:FIELD 1, NOM(I) TO $8,:NEXT I

4) Il semblerait, au vu du code en FC2E, que le même nom de champ puisse être défini deux fois pour deux champs différents au sein d'une même fiche. En fait, ceci a pour conséquence d'effacer les informations de la première définition et les data contenus dans le premier champ qui deviennent donc inaccessibles. Ce phénomène semble voulu, mais il serait peut-être mieux de reprogrammer cette partie de manière à prévenir ("DUPLICATE FIELD).

5) Il est également possible d'avoir le même "nom_de_champ(index)" dans plusieurs fichiers. Cependant, peut-être à cause d'une bogue du Sédoric, lors d'un transfert de ou vers un champ, grâce aux commandes LSET ou RSET, seul le 1er "nom_de_champ(index)" est accessible. Ceci est dû au fait que Sédoric ne compare pas le **NL** pour lequel le "nom_de_champ(index)" a été définis avec le **NL** courant. La vérification du **NL** et du "nom_de_champ(index)" peut être obtenue en changeant un octet de Sédoric: remplacer BVC F548 (50 20) par BVC F54B (50 23) en F526.

En temps normal, il n'y a pas de vérification du **NL** il en résulte que la première occurrence du "nom_de_champ(index)" rencontrée est acceptée comme la bonne, sans tenir compte du **NL**. Si le même "nom_de_champ(index)" est utilisé par plusieurs **NL**, alors l'adressage trouvé risque de ne pas être le bon. Après modification de Sédoric en F526, le **NL** courant sera utilisé pour la vérification et les deux instructions en F5CE et F5D1 seront redondantes.

Cependant, avec le code en vigueur (Sédoric non modifié), si tous les "noms_de_champ(index)" sont différents, on peut obtenir sans problème le champ de la fiche courante quel que soit le fichier, à n'importe quel moment, du moment que le fichier est ouvert et que la bonne fiche est en place. Il en sera de même pour les commandes LSET et RSET.

LSET nom_de_champ,expression et **RSET nom_de_champ,expression**, cette commande transfère une expression ou une variable dans le champ indiqué. Pour LSET, les valeurs alphanumériques sont placées à gauche dans le champ et justifiée à droite avec des espaces (ou tronquées si trop longues). Pour RSET, les valeurs alphanumériques sont placées à droite dans le champ et justifiée à gauche avec des espaces (ou tronquées à gauche si trop longues).

Il faut utiliser la commande TAKE pour mettre à jour le **NL** et le n° de fiche. Le nom_de_champ(index), qui doit avoir été défini pour le **NL** courant (sinon "UNKNOWN FIELD NAME ERROR"), est décodé dans le "General Field Buffer", qui se trouve en C076/C07F. L'expression doit correspondre au type du champ indiqué (alphanumérique, réel, entier ou octet), sinon "TYPE MISMATCH

ERROR".

Lors du transfert d'une expression alphanumérique vers un champ, celui-ci est d'abord effacé avec des espaces puis l'expression est copiée dans le champ de gauche à droite ou de droite à gauche, en limitant le nombre de caractères copiés à la longueur du champ. Pour les autres types de champs, la longueur de l'objet étant définie par le type LSET et RSET donnent le même résultat!

Non documenté: LSET signifie "left set", c'est à dire "placer à gauche" et RSET signifie "right set", c'est à dire "placer à droite". Ceci n'a de sens que pour les expressions alphanumériques!

nom_de_champ > variable, lit le champ de nom spécifié et en affecte la valeur à la variable BASIC indiquée. Il faut d'abord utiliser la commande TAKE pour mettre à jour le n° de fiche et le **N**L. Le nom_de_champ (index), qui doit avoir été défini pour le **NL** courant (sinon "UNKNOWN FIELD NAME ERROR", est décodé dans le "General Field Buffer" qui se trouve en C076/C07F. Le champ et la variable doivent être du même type (alphanumérique ou numérique) et de la même longueur, sinon "TYPE MISMATCH ERROR". En ce qui concerne le type numérique, il y a tolérance entre les types réel, entier ou octet.

Non ou mal documenté: cette commande est mal sécurisée: elle ne peut pas être utilisée avec un fichier de type **S**, mais cela n'est pas vérifié. De plus, il n'est pas testé si un fichier **R** à accès direct est ouvert et qu'une fiche a été chargée ou si un fichier d'accès **D**isque est ouvert et qu'un secteur a été chargé. La validité de la fiche n'est pas vérifiée et il se peut en fait qu'elle ne contienne aucun data valable.

CLOSE (NL)(,NL)(,NL) etc. Libère le ou les **NL** indiqués (et tous les **NL** en absence d'indication), ferme le ou les fichier(s) correspondant(s) et récupère en absence d'indication le ou les buffer(s) devenu(s) inutilisé(s). Lorsqu'un **NL** indiqué n'est pas attribué, on obtient une "FILE NOT OPEN ERROR".

Pour chaque **NL** devant être "fermé", l'octet de poids fort de l'offset correspondant à ce **NL** dans la "Table **NL**" est remis à zéro pour indiquer que ce **NL** est désormais libre et que le fichier correspondant est fermé.

Chaque champ relatif au **NL** devant être "fermé" est libéré en plaçant un zéro au début de l'entrée correspondante dans le "Field Buffer", afin d'indiquer que cet emplacement est désormais disponible. Cette disponibilité sera utilisée lors d'une prochaine utilisation de la commande FIELD.

Par contre, le "Channel Buffer" correspondant au **NL** (#121 octets de long au minimum, plus si le fichier comporte plusieurs secteurs descripteurs ne sera pas de nouveau utilisable, même si le **NL** demandé est identique), d'où une perte de place (rapide si l'on ouvre et ferme beaucoup de fichiers) pouvant occasionner une "OUT OF MEMORY ERROR". Il ne semble pas trop difficile de reprogrammer le Sedoric pour que dorénavant la partie la plus haute de **FI** soit redescendue en mémoire lors de la fermeture d'un fichier.

Informations non documentées: attention à la bogue: ne pas utiliser un CLOSE sans paramètre (fermeture de tous les fichiers en mémoire) en milieu de programme (**FI** devient inutilisable). Même si la fermeture a bien lieu, celle-ci s'effectue à partir du **NL** 99 (#63) et non 63 (#3F). Cette erreur de programmation peut être facilement résolue en remplaçant LDA #63 par LDA #3F en FBA3.

&(NL) et **&(-NL)**. Attention, les paramètres n'indiquent pas une option facultative, mais sont obligatoires. Cette commande retourne des informations qui diffèrent selon le signe du paramètre et le type de fichier. Le paramètre entre parenthèse est décodé par la Rom du BASIC et placé en virgule flottante dans ACC1 (D0/D4). Sedoric vérifie que le paramètre se situe bien entre -63 et +63, puisqu'il s'agit d'un **NL** (qui doit être attribué, sinon "FILE NOT OPEN ERROR"). Si le **NL** est attribué à un fichier de type **D**, il en résulte une "FILE TYPE MISMATCH ERROR" (commande non utilisable pour les fichier de type **D**).

Informations non (ou très mal) documentés:

Pour les fichiers de type **R**, cette commande retourne le nombre de fiches si **&(-NL)** ou la longueur de fiche si **&(-NL)**.

Pour les fichiers de type **S**, cette commande retourne -1 (vrai) dans tous les cas (**+/-NL**) si la fin du fichier est atteinte et si ce n'est pas le cas, retourne soit 0 (false) si &(**+NL**), soit le type d'enregistrement, si **&(-NL**). C'est le contraire de ce qui est indiqué dans le manuel, page 91 (bogue du manuel).

Nous espérons que ces informations vous ont paru suffisamment claires et vous donnons rendez-vous dans un prochain article...