Metro Cross

par Simon Guyart & André Chéramy

Le mois dernier, nous vous avons présenté la petite demo-animation préparée par Simon pour la visu de janvier. Cette animation était dépendante du DOS, car elle utilisait la commande MOVE de Sedoric. Rien de plus simple que d'écrire l'équivalant de cette commande en langage machine. C'est ce que nous avons fait et nous vous proposons ici une version K7. Mais plus intéressant encore, il est maintenant possible d'envisager une plus grande animation, avec une programmation 100% langage machine, donc beaucoup plus rapide que le BASIC.

Routine MOVE : Principe

On parle de déplacement, plutôt que de copie, car dans certains cas, le bloc mémoire source est partiellement écrasé par le bloc cible. Mais si les zones source et cible sont distantes, MOVE réalise en fait une copie. Toutefois, une routine MOVE, qui se voudrait universelle, devrait être composée de deux routines : la première pour déplacer le bloc mémoire vers les adresses basses et l'autre vers les adresses hautes.

En effet, si on copie, vers les adresses basses, un bloc mémoire à partir du premier octet de ce bloc, il n'y a jamais de problème, même si le «déplacement» n'est que d'un octet et que le bloc cible écrase au fur et à mesure le bloc source.

Mais il en est autrement si on déplace un bloc mémoire vers le haut. En effet, si la zone cible est proche du bloc source (en fonction de la longueur du bloc à déplacer), on court le risque d'écraser la fin de la zone source avant de l'avoir copiée. Dans ce cas, il faut donc commencer à copier **à partir de la fin** du bloc source, **vers la fin** du bloc cible, à l'envers en quelque sorte.

Une routine MOVE universelle doit donc en premier lieu déterminer si le déplacement est à faire «vers le bas» ou «vers le haut». Dans notre cas particulier, il s'agira toujours de copier (et surtout pas d'écraser) de la mémoire basse (située sous les caractères) vers la zone de définition des caractères. Pour être même plus précis, dans l'exemple présent la copie se fait de la zone #AF80 et suiv. vers la zone #B708 et suiv. (définition des lettres « a » à « l »).

Il n'était donc pas nécessaire que notre routine MOVE soit universelle et c'est donc une routine simplifiée que nous vous proposons. De plus, dans notre exemple, les zones source et cible étant distinctes, il n'y a aucun risque d'écrasement et nous aurions pu effectuer la copie «normalement», c'est à dire depuis le début du bloc.

Mais la routine que nous vous proposons présente la rare qualité d'être directement utilisable n'importe où en mémoire (il n'y a que des adressages relatifs) et donc d'être immédiatement re-utilisable. Pour que vous puissiez la re-utiliser dans tous vos propres programmes nécessitant un déplacement vers le haut, nous effectuerons donc quand même la copie en commençant par la fin du bloc (pour quelques octets de plus, comme dirait Sergio Leone !).

La Routine MOVHAUT

Elle occupe 85 octets de #B473 à #B4C7 (mais pourrait être chargée n'importe où en mémoire). L'adresse de début de bloc source doit être DOKée à l'adresse #00, celle de fin de bloc source en #02 et enfin celle de début de bloc cible en #04. Pour simplifier notre routine MOVHAUT, nous aurions pu DOKer directement les valeurs utilisées en pratique par la routine, c'est à dire adresses de fin de

bloc source et de fin de bloc cible, ainsi que nombre d'octets à copier. Mais nous avons préféré garder la syntaxe qui est universellement utilisée par les routines MOVE. La routine devra donc se débrouiller pour faire les calculs. Voici donc le code de MOVHAUT :

```
B473-   A0 00       LDY #$00     pointeur de lecture/écriture
B475-   D8          CLD
B476-   38          SEC          prépare soustraction
B477-   A5 02       LDA $02      calcule adr fin - adr deb
B479-   E5 00       SBC $00      c'est à dire
B47B-   85 02       STA $02      le nombre d'octets
B47D-   A5 03       LDA $03      à copier (-1)
B47F-   E5 01       SBC $01      et stocke le résultat à la
B481-   85 03       STA $03      place de adr fin en 02/03
B483-   18          CLC          prépare addition
B484-   A5 00       LDA $00      calcule l'adr
B486-   65 02       ADC $02      de fin de bloc source
B488-   85 00       STA $00      soit adr source
B48A-   A5 01       LDA $01      + nombre octet (-1)
B48C-   65 03       ADC $03      et place résultat en 00/01
B48E-   85 01       STA $01
B490-   18          CLC          prépare addition
B491-   A5 04       LDA $04      calcule l'adr
B493-   65 02       ADC $02      de fin de bloc cible
B495-   85 04       STA $04      soit adr cible
B497-   A5 05       LDA $05      + nombre octet (-1)
B499-   65 03       ADC $03      et place résultat en 04/05
B49B-   85 05       STA $05
B49D-   E6 02       INC $02      corrige le nombre d'octets
B49F-   D0 02       BNE $B4A3
B4A1-   E6 03       INC $03      en ajoutant 1
B4A3-   B1 00       LDA ($00),Y  lit un octet dans source
B4A5-   91 04       STA ($04),Y  et le copie dans cible
B4A7-   A5 00       LDA $00      décrémente
B4A9-   D0 02       BNE $B4AD    l'adr de fin
B4AB-   C6 01       DEC $01      de bloc source
B4AD-   C6 00       DEC $00      en 00/01
B4AF-   A5 04       LDA $04      décrémente
B4B1-   D0 02       BNE $B4B5    l'adr de fin
B4B3-   C6 05       DEC $05      de bloc cible
B4B5-   C6 04       DEC $04      en 04/05
B4B7-   A5 02       LDA $02      décrémente
B4B9-   D0 02       BNE $B4BD    le nombre d'octets
B4BB-   C6 03       DEC $03      restant à copier
B4BD-   C6 02       DEC $02
B4BF-   A5 03       LDA $03      teste s'il en reste
B4C1-   D0 E0       BNE $B4A3    si oui, reboucle
B4C3-   A5 02       LDA $02
B4C5-   D0 DC       BNE $B4A3
B4C7-   60          RTS          si non, termine
```

Le petit programme BASIC du mois dernier doit être modifié comme suit : la ligne

```
MOVE 44928+I*12*8,44928+I*12*8+96,46856
```

doit simplement être remplacée par la ligne

```
DOKE#0,44928+I*12*8 : DOKE#2,44928+I*12*8+96 : DOKE#4,46856 : CALL#B473
```

Pour faire une version K7 de cette démo, il faudra aussi remplacer la commande Sedoric de chargement direct ANIMDAT par CLOAD«ANIMDAT» : CLOAD«MOVHAUT». La démo comportera donc maintenant trois fichiers au lieu de deux : Le programme BASIC, la redéfinition des caractères ANIMDAT et la routine MOVHAUT.