Visitons la Rom Monitoring

(2e partie) par André Chéramy et Claude Sittler

M) Ancienne série de 7 routines pour CSAVE en E607-E6C8. A la place de ces routines on trouve maintenant :

a) Nouvelle commande DUMP en E607-E62F.

Syntaxe : DUMP#xxxx (adresse hexadécimale) ou DUMPxxxx (adresse décimale)

A partir de l'adresse indiquée, affiche les octets 8 par 8 avec conversion Ascii. Le défilement peut être interrompu et repris avec la touche 'espace'. Sortie avec la touche 'ESC'.

```
E607- 20 53 E8  JSR E853   évalue adr, la garde en 33-34
E60A- 20 5B F8  JSR F85B   va à la ligne, affiche cette
      adresse en hexadécimal et affiche un espace
E60D- A0 00     LDY #00    index de lecture
E60F- B1 33     LDA(33),Y  lit dans l'accumulateur A
      l'octet pointé par 33-34 + index
E611- 20 34 FF  JSR FF34   et l'affiche en hexadécimal
E614- 20 D4 CC  JSR CCD4   affiche un espace
E617- C8        INY        incrémente pour octet suivant
E618- C0 08     CPY #08    maximum 8 octets par ligne
E61A- D0 F3     BNE E60F   reboucle si limite pas atteinte
E61C- A2 08     LDX #08    si sortie hexa terminée,
E61E- A0 00     LDY #00    prépare une boucle et un
      index pour relire les 8 octetset les afficher en Ascii
E620- B1 33     LDA(33),Y  lit dans l'accumulateur A
      l'octet pointé par 33-34 + index
E622- 20 4C FF  JSR FF4C   affiche le caractère Ascii
      correspondant ou un espace
E625- 20 89 F8  JSR F889   incrémente adr en 33-34
E628- CA        DEX        suivant
E629- D0 F3     BNE E61E   reboucle 8 fois
E62B- 20 0A E9  JSR E90A   teste si 'espace' ou 'ESC' et
      gère pause ou fin
E62E- D0 DA     BNE E60A   sinon, dumpe 'à l'infini'
```

b) Nouvelle commande SEARCH en E630-E66B.

Syntaxe : SEARCH A#xxxx,E#xxxx,#AA,#BB,#CC etc.

Recherche les octets #AA, #BB, #CC etc. entre les adresses indiquées et affiche les adresses des occurrences trouvées. Remarque : On peut chercher une suite comportant jusqu'à 8 octets au maximum. Les adresses et les valeurs peuvent être indiqués en hexadécimal (avec un '#') ou en décimal.

```
E630- 20 E8 00  JSR 00E8   relit octet à TXTPTR
E633- 20 92 EA  JSR EA92   Demande A+adr (met en 02-
      03) et E+adr (met en 33-34)
E636- 20 65 D0  JSR D065   demande «,» lit octet suivant
E639- A2 08     LDX #08    pour chaîne de 8 octets maxi
E63B- 86 7F     STX 7F     sauve la valeur actuelle de X
E63D- 20 53 FE  JSR FE53   lit dans A octet à TXTPTR
E640- A6 7F     LDX 7F     récupère index X
E642- 95 76     STA 76,X   et sauve le résultat dans table
E644- 20 E8 00  JSR 00E8   relit octet à TXTPTR
E647- F0 06     BEQ E64F   branche s'il n'y en a plus
E649- CA        DEX        suivant
E64A- 20 65 D0  JSR D065   demande «,» et lit suivant
E64D- D0 EC     BNE E63B   reboucle si Y a pas atteint 0
E64F- A0 00     LDY #00    si fini, commence recherche
E651- A2 08     LDX #08    pour 8 octets maxi
E653- B1 02     LDA (02),Y lit à partir de Axxxx
E655- D5 76     CMP 76,X   compare avec le 1er octet
E657- D0 0D     BNE E666   continue si différent
E659- C8        INY        si identique, indexe suivant
E65A- CA        DEX        décrémente nbre à comparer
E65B- E4 7F     CPX 7F     teste si nombre octets atteint
E65D- B0 F4     BCS E653   si non, reboucle
E65F- A5 02     LDA 02     si oui, on a trouvé
E661- A4 03     LDY 03     à l'adresse AY
E663- 20 92 F8  JSR F892   va la ligne puis affiche en
      hexadécimal le nombre AY
E666- 20 84 E5  JSR E584   teste si adr02-03 < adr33-34
E669- 90 E4     BCC E64F   si oui, reboucle
E66B- 60        RTS        si non, retourne
```

c) Nouvelle commande MOD en E66C-E681.

Syntaxe MOD#xxxx (adresse hexadécimale) ou MODxxxx (adresse décimale). Edite la valeur des octets à partir de l'adresse indiquée. L'utilisation de cette commande est délicate et demande beaucoup de rigueur. La commande affiche en hexadécimal l'adresse, puis l'octet à modifier suivit d'un '?'. Il attend l'une des 3 réponses suivantes :

-Un 'espace', auquel cas il garde la valeur actuelle et passe à l'adresse suivante.

-Trois digits maxi :

soit '#'+1ou 2 digits hexa de 0 à FF suivis de 'Return'.

soit de 1 à 3 digits décimaux de 0 à 255 suivis de 'Return'.

-La touche 'ESC' pour terminer.

Remarques:

1) Le texte n'est pas valide, sauf sous forme de code Ascii (65 ou #41 pour la lettre 'A' etc.).

2) La touche 'Return' sert à valider, un 'Return' tout seul génère une sortie avec erreur, mais les entrées précédentes restent valides. Utilisez plutôt la touche 'ESC' pour sortir.

3) Le CTRL/A est actif. Comme d'habitude, il peut s'utiliser en conjonction avec les flèches pour recopier des caractères déjà présents à l'écran. Mais il vaut mieux ne pas s'en servir, car c'est plus compliqué que de retaper 3 digits et peut facilement produire des erreurs. En effet, le programme est très peu tolérant. Certes, il accepte des choses comme '# 22' pour #22, mais le plus souvent il avale sans rien dire et inscrit n'importe quoi en mémoire.

4) Le fait que la commande MOD affiche l'octet à modifier en hexadécimal sans '#', mais réclame un '#' devant une entrée hexadécimale est troublant. Attention à ne pas entrer du décimal en pensant fournir de l'hexadécimal ! En cas de doute, vérifiez avec un DUMP.

5) On ne peut entrer qu'un octet à la fois. Si dans le feu de l'action, vous en tapez plusieurs, corrigez si possible avant de valider. En général seul le premier d'une série est pris en compte, mais il existe beaucoup de conditions d'erreurs. En cas de doute, vérifiez avec un DUMP.

6) Les octets, qu'ils soient entrés en décimal ou en hexadécimal, peuvent être 'justifiés' avec un ou deux zéros par devant. Par exemple '#4' et '#04' sont équivalents, de même pour '4', '04' ou '004'.

E66C- 20 53 E8 JSR E853 évalue adr et la met en 33-34
E66F- 20 5B F8 JSR **F85B** va à la ligne, affiche cette adresse en hexadécimal et affiche un espace
E672- 20 D4 CC JSR CCD4 affiche un autre espace
E675- A0 00 LDY #00 index de lecture
E677- B1 33 LDA (33),Y lit l'octet actuel au pointeur
E679- 20 34 FF JSR **FF34** et l'affiche en hexadécimal
E67C- 20 D7 CC JSR CCD7 affiche un «?»
E67F- 4C 93 C6 JMP **C693** suite pour saisir un octet

d) Nouvelle routine «Copie les octets du tampon clavier à l'adresse pointée en 33-34". Cette routine est appelée par la commande MOD via la routine C693.

E682- E8 INX incrémente le nombre de caractères placés dans le tampon clavier
E683- 86 E9 STX E9 et initialise TXTPTR (E9-EA)
E685- 20 53 FE JSR **FE53** lit un octet dans le tampon clavier selon TXTPTR
E688- A0 00 LDY #00 et l'écrit à l'adresse
E68A- 91 33 STA (33),Y pointée en 33-34
E68C- 20 89 F8 JSR **F889** incrémente adresse en 33-34 et
E68F- 4C 6F E6 JMP E66F reboucle dans la Cde MOD

e) Nouvelle routine «Lit 3 entiers à TXTPTR, destinés aux registres A, X, Y et les stocke aux adresses 06, 07 et 08». Cette routine est appelée par la commande CALL via la routine en FE67. Si la nouvelle commande CALL a été utilisée sans indiquer de valeur pour A, X et Y cette routine retournera à la routine appelante FE67 sans rien faire et par la suite les registre A, X et Y seront faussement initialisés !

E692- A9 03 LDA #03 pour 3 tours
E694- 85 09 STA 09
E696- 20 E8 00 JSR 00E8 relit l'octet à TXTPTR
E699- F0 28 BEQ E6C3 termine si octet nul
E69B- 20 65 D0 JSR D065 demande ',' lit caractère suiv.
E69E- C9 41 CMP #41 est-ce 'A' ?
E6A0- F0 18 BEQ E6BA oui continue en E6BA
E6A2- C9 58 CMP #58 est-ce 'X' ?
E6A4- F0 0C BEQ E6B2 oui continue en E6B2
E6A6- C9 59 CMP #59 est-ce 'Y' ?
E6A8- D0 1A BNE E6C4 non, «SYNTAX ERROR»
E6AA- 20 5C FE JSR **FE5C** cas de 'Y'
E6AD- 85 08 STA 08 stocke à l'adresse 08
E6AF- 4C BF E6 JMP E6BF
E6B2- 20 5C FE JSR **FE5C** cas de 'X'
E6B5- 85 07 STA 07 stocke à l'adresse 07
E6B7- 4C BF E6 JMP E6BF
E6BA- 20 5C FE JSR **FE5C** cas de 'A'
E6BD- 85 06 STA 06 stocke à l'adresse 06
E6BF- C6 09 DEC 09 décrémente nombre de tours
E6C1- D0 D3 BNE E696 et reboucle si pas fini
E6C3- 60 RTS retourne
E6C4- 4C 70 D0 JMP D070 «SYNTAX ERROR»
E6C7- EA NOP
E6C8- EA NOP

N) Modifications de la Cde CLOAD en E85B-E908.

La modification porte sur la fin de la commande de E8D3 à E909 et déborde donc d'un octet sur la commande suivante CSAVE, qui a été totalement supprimée. Elle remplace le test sur la vérification des erreurs de parité, qui de toute façon ne fonctionnait pas, par un affichage en hexadécimal les adresses de début et de fin du fichier chargé et corrige la bogue «ERRORS FOUND». Sinon le code est identique au précédent, quoique dans un ordre différent.

E8D3- 20 A0 FC JSR **FCA0** affiche en hexadécimal les adresses de début et de fin
E8D6- AD B1 02 LDA 02B1 erreurs pendant la lecture ?
E8D9- F0 07 BEQ E8E2 non, saute 3 instructions suiv.
E8DB- A9 27 LDA #27 oui, AY pointe la chaîne
E8DD- A0 E5 LDY #E5 «ERRORS FOUND»
E8DF- 20 B0 CC JSR CCB0 l'imprime et continue
E8E2- AD AE 02 LDA 02AE type de fichier (0 si Basic)
E8E5- F0 08 BEQ E8EF si Basic suite en E8EF
E8E7- AD AD 02 LDA 02AD sinon, teste si Auto ou pas
E8EA- F0 E6 BEQ E8D2 si 0, pas Auto, simple RTS
E8EC- 6C A9 02 JMP (02A9) si non nul, c'est Auto, exécute le programme à son adresse de début
E8EF- AE AB 02 LDX 02AB octet LL de l'adresse de fin
E8F2- AD AC 02 LDA 02AC octet HH de l'adresse de fin
E8F5- 86 9C STX 9C met à jour le pointeur
E8F7- 85 9D STA 9D de fin de Basic
E8F9- 20 5F C5 JSR C55F restaure les liens des lignes
E8FC- AD AD 02 LDA 02AD teste si Auto ou pas
E8FF- F0 03 BEQ F904 pas Auto, saute l'instr. suiv.
E901- 4C 08 C7 JMP C708 si Auto place TXTPTR au début du programme, effectue un CLEAR et retourne au programme appelant.
E904- 20 08 C7 JSR C708 si pas Auto, place TXTPTR
E907- 4C A8 C4 JMP C4A8 au début du programme, effectue un CLEAR et retourne à l'interpréteur

O) Ancienne commande CSAVE située en E909-E93C.

a) Nouvelle routine «Teste si 'espace' ou 'ESC' et gère pause ou fin». Appelée par DES et DUMP. Retourne immédiatement au listing si ni 'ESC' ni 'espace'. Si 'ESC', termine le DES ou le DUMP. Si 'espace' attend une autre touche (sauf 'ESC') pour reprendre l'affichage.

E90A- 20 78 EB JSR EB78 teste si une touche a été pressée et met le code Ascii dans l'accumulateur A
E90D- C9 1B CMP #1B est-ce 'ESC' ?
E90F- F0 F6 BEQ **E907** si oui, retour à l'interpréteur
E911- C9 20 CMP #20 est-ce 'espace' ?
E913- D0 09 BNE E91E si pas 'espace', simple RTS
E915- 20 78 EB JSR EB78 teste si une touche à été pressée et si c'est le cas met l'indicateur N à 1 (toute touche sera valide pour reprendre le listing sauf ESC)
E918- 10 FB BPL E915 si pas de touche, reboucle
E91A- C9 1B CMP #1B si touche, est-ce 'ESC' ?
E91C- F0 E9 BEQ **E907** si oui, retour à l'interpréteur
E91E- 60 RTS si pas ESC, reprend le listing

b) Entrée de la nouvelle commande CLEAN.

Syntaxe : CLEAN A#xxxx,E#xxxx,#XX

Met des octets #XX entre les deux adresses. Les valeurs peuvent être données en décimal (sans le '#' bien sûr).

E91F- 20 E8 00 JSR 00E8 relit l'octet à TXTPTR
E922- 20 92 EA JSR **EA92** Demande A+adr (met en 02-03) et E+adr (met en 33-34)
E925- 20 65 D0 JSR D065 demande «,» et lit suivant
E928- 20 53 FE JSR **FE53** lit un octet à TXTPTR
E92B- A5 D4 LDA D4 le récupère
E92D- A0 00 LDY #00 initialise un index
E92F- 91 02 STA (02),Y l'écrit à adresse en 02-03
E931- 20 84 E5 JSR **E584** incrémente 02-03 et teste si fin
E934- 90 F5 BCC E92B si pas fini, reboucle
E936- 60 RTS si fini, retourne
E937- 6 x EA 6x NOP neutralise le reste de la place

à suivre...