RipDOS V2.9 (2)

XtraDOS

by Jim Polmear et André Chéramy

J'ai déjà évoqué dans l'article précédent la possibilité, introduite par Jim, d'étendre les commandes de RipDOS. Le système mis en place par Jim est d'une simplicité enfantine (pour l'usager),mais d'une efficacité redoutable !

Sans vouloir traduire mot à mot le texte ci-contre de Jim, en voici les principales lignes. Lorsque l'interpréteur du DOS n'a pas trouvé le nom cherché dans la table des commandes normales, il poursuit sa recherche dans un table supplémentaire, située de **D000 à D0BF** (les noms sont **séparés par le drapeau #00**), jusqu'à ce qu'il rencontre le **#FF de fin de table**. *[Note : En absence de table supplémentaire (RipDOS de base), il y a un #FF en D000, au début de la table.]* Si le nom n'a pas été trouvé, l'interpréteur tente de charger le fichier 'NOM.COM' et en absence de celui-ci génère une 'Invalid Filename Error'. Si le nom est trouvé dans la table, l'interpréteur va lire l'adresse d'exécution ayant le même n° d'ordre dans la table suivante, située de **D0C0 à D0FF**. *[Note : Il s'agit en fait de l'adresse d'exécution '-1'.]* Enfin, il lance l'exécution de la commande du code, situé à l'adresse correspondante dans la zone **D100 à DFFF.**

Voilà, c'est tout ! Il suffit de sauver la zone allant de D000 au RTS de la dernière commande, sous un nom du genre XTRA.DOS Il est ainsi possible d'insérer 64 nouvelles commandes !

Pour utiliser ce DOS complémentaire, il suffit de charger le fichier XTRA.DOS après le boot de RipDOS. Ceci peut se faire automatiquement avec le fichier BOOTUP.COM qui est lancé automatiquement au boot.

Jim a développé ainsi plusieurs DOS complémentaires : XTRA01.DOS, XTRA02.DOS et XTRA03.DOS, qui regorgent de nouvelles commandes dont certaines feraient le bonheur de bien des Oriciens. Les plus originales de ces commandes feront l'objet de plusieurs autres articles.

Voulant tester par moi-même de la viabilité et de la facilité du système de Jim, je me suis décidé à créer un XTRA.DOS de démonstration (sans intérêt pratique autre que de voir comment ça marche en pratique). Je pense à une commande nommée SALUT, qui affichera le message 'Salut les gars !!' sur la première ligne de l'écran texte normal, d'adresse BBA8.

Pour ce faire, j'ai utilisé sous Sedoric un assembleur développé par Jim et qu'il m'avait envoyé dans l'image ASSEMBLE.DSK

En voici les grandes lignes d'utilisation pratique, pour ma démonstration de commande supplémentaire :

1) Je boote avec RipDOS et charge l'assembleur avec !LOAD'ASSEMB.LER' (une interface en Basic commançant à la ligne 30000).

2) Je tape le listing source suivant :

New Feature of RipDOS

A routine has been added so that an additional table of commands and their code can be installed in an unused area of memory from #D000 to #DFFF.

The facility, which allows a new set of commands to be added, works through the normal command interpreter routine in the DOS.

If an unrecognised command is encountered, the routine looks to see if there is another look up table in page #D000. If it finds a match there, it will run the routine. If it does not find a match, it will come up with an error message.

The command word table MUST start at #D000 and each entry must be separated with a null (0). Don't forget that some words may be partly or completely tokenised. For example, FORGET will be represented by #8D (FOR) #BE (GET) #00 (null to separate).

#D000 to #D0BF is available for the new table of words, which MUST end with #FF after the null of the last entry.

Memory #D0C0 to D0FF is reserved for the corresponding table of addresses for the new routines. Each address is 2 bytes long with the low byte first. There must be a 2 byte entry for each new command word in the first part of the table. Each address must be 1 less than the starting address for its routine, since 1 is added when the routine is called using the RTS instruction.

Memory #D100 to DFFF is available for the routines themselves. This is in an unused area of DOS memory, and provides huge flexibility for providing your own set of additional utilities without using RAM.

I really do need an example to show how this works!

Disassembly of interpreter :

Match primary commands

```
E012 BA       TSX        Entry point
                         into DOS
E013 8E 07 C0 STX C007
E016 A9 00    LDA #00
E018 8D FF 04 STA 04FF
E01B A6 EA    LDX EA
E01D D0 03    BNE E022
E01F 8E FD 04 STX 04FD
E022 A2 FF    LDX #FF    Try to match
                         command word
E024 8E 40 C1 STX C140   Command word
                         counter
```

```
100 DA$ "SALUT"
110 DA# 0
120 DAB 186,#FF
200 DA# 11,D1
210 DAB 62,#00
310 DA$ "Salut les gars !!"
320 DA# 0
330 LDY #00
339 LBL12
340 LDA D100,Y
350 BEQ L13
360 STA BBAA,Y
370 INY
380 BNE L12
390 LBL13
400 RTS
410 END
```

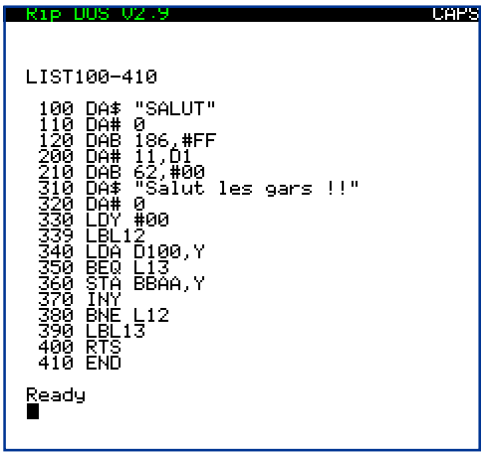
Comme vous le devinez, Jim utilise un listing 'Basic' comme source pour le code d'assemblage. Ce programme source ne sera bien-sûr pas lu par l'interpréteur Basic, mais par l'assembleur de Jim.

Sans vouloir entrer dans le détail, voici quelques remarques sur ce code assembleur. Aux lignes 100 etc., le nom de la nouvelle commande, suivie de #00 (drapeau de fin de nom, puis de #FF (drapeau de fin de liste). En fait j'ai rempli le reste de la zone de table des commandes D000 à D0BF avec des #FF (par facilité).

Aux lignes 200 etc., l'adresse '-1' d'exécution de la commande : D111 (avec l'octet de poids faible en tête). Nous verrons plus loin que premier octet du code exécutable se trouve en D112. Je rempli le reste de la zone de table d'adresses DOC0 à D0FF avec 62 fois #00.

A la ligne 300 commence le programme, ou plus précisément les données du programme, soit ici le message "Salut les gars !!", suivit de #00 (drapeau de fin de message).

Puis à la ligne 320 commence le code exécutable. J'ai oublié de mettre une étiquette devant cette ligne, pour que l'assembleur m'indique cette adresse. Mais le programme de Jim possède aussi un désassembleur que j'ai essayé et qui m'a renseigné : L'adresse est D112.




```
E027 A0 FF    LDY #FF
E029 EE 40 C1 INC C140
E02C E8       INX        X counts along the
                         lookup table
E02D C8       INY        Y counts characters
                         in the command word
E02E BD 40 FF LDA FF40,X
E031 C9 FF    CMP #FF
E033 F0 28    BEQ E05D   Exit if end of table
                         reached with no match
                         found
E035 48       PHA
E036 68       PLA
E037 F0 0C    BEQ E045   Accept command word
                         when null reached
E039 D1 E9    CMP (E9),Y
E03B F0 EF    BEQ E02C   Accept the current
                         character and go back
                         for another
E03D E8       INX        otherwise move on to
                         try the next word in
                         the table
E03E BD 40 FF LDA FF40,X
E041 D0 FA    BNE E03D
E043 F0 E2    BEQ E027
E045 98       TYA        Command word accepted
E046 18       CLC        Move text pointer on to
                         the end of command word
E047 65 E9    ADC E9       (Y = number of
                         characters in word)
E049 85 E9    STA E9
E04B 90 02    BCC E04F
E04D E6 EA    INC EA
E04F AD 40 C1 LDA C140   Retrieve command word
                         counter
E052 0A       ASL        and double it as in-
                         dex for address
E053 AA       TAX
E054 BD C1 FF LDA FFC1,X Fetch the address (-
                         1) for the command word
E057 48       PHA        Push it onto the
                         stack
E058 BD C0 FF LDA FFC0,X
E05B 48       PHA
E05C 60       RTS        and JMP to it via the
                         RTS
```

Match secondary commands

```
E05D A2 FF    LDX #FF    Same process as above,
E05F 8E 40 C1 STX C140   but looking in a se-
                         cond command table
E062 A0 FF    LDY #FF    held at #D000 onwards
E064 EE 40 C1 INC C140
E067 E8       INX
E068 C8       INY
E069 BD 00 D0 LDA D000,X Secondary command
                         word table starts at
                         #D000
E06C C9 FF    CMP #FF    Exit if end of table
                         reached with no match
E06E F0 28    BEQ E098   (the start up routine
                         puts #FF into #D000
E070 48       PHA        so there will be an
                         immediate exit
E071 68       PLA        if extra commands
                         have not been added)
```

```
                                                        CAPS

Ready
!SAVE"SALUT.C",A#A200,E#A31F
Saving.. SALUT  .C

Ready
!LOAD"SALUT.C",A#D000,D
Loading.. SALUT  .C
D000 D11F

Ready
!SAVE"XTRA.DOS",A#D000,E#D11F
Saving.. XTRA   .DOS

Ready
```

Je ne commenterai pas plus le code très simple de ce programme, que les lecteurs de la rubrique 'Initiation Assembleur' reconnaîtront.

3) Je sauve ce travail avec un !SAVE'SALUT.S', ainsi que le listing ci-dessus avec LLIST 100-410 et avec une recopie d'écran F12 (voir figure jointe).

4) Un RUN30000 permet de lancer l'assembleur, l'option '3' permet d'assembler le code, l'option '6' de lister les labels (je note que le code a été assemblé de A200 à A31F), l'option '9' de sortir et de sauver ce code avec !SAVE'SALUT.C',A#A200,E#A31F

5) Je mets en place ce code dans la Ram Overlay avec !LOAD'SALUT.C',A#D000 et le resauve avec les bonnes adresses : !SAVE'XTRA.DOS',A#D000,E#D11F Nouvelle recopie d'écran avec F12 (voir figure jointe).

6) Je reboote (F6), charge le dos complémentaire avec !LOAD'XTRA.DOS' et tente un !SALUT et le message s'affiche bien là où il faut (dernière recopie d'écran). Il n'y a pas de miracle, mais ça fait bien plaisir !

La prochaine fois nous aborderons les nouvelles commandes crée par Jim...

```
Rip DOS V2.9                                            CAPS

!LOAD"XTRA.DOS",D
Loading.. XTRA  .DOS
D000 D11F

Ready
```

```
Rip DOS V2.9                                            CAPS
Salut les gars !!

!LOAD"XTRA.DOS",D
Loading.. XTRA  .DOS
D000 D11F

Ready
!SALUT

Ready
```

```
E072 F0 0C    BEQ E080   Accept command word
                         when null reached
E074 D1 E9    CMP (E9),Y
E076 F0 EF    BEQ E067   Accept the current
                         character and go back
                         for another
E078 E8       INX        otherwise move on to
                         try the next word in
                         the table
E079 BD 00 D0 LDA D000,X
E07C D0 FA    BNE E078
E07E F0 E2    BEQ E062
E080 98       TYA        Secondary command
                         word accepted
E081 18       CLC        Move text pointer on to
                         the end of command word
E082 65 E9    ADC E9
E084 85 E9    STA E9
E086 90 02    BCC E08A
E088 E6 EA    INC EA
E08A AD 40 C1 LDA C140   Retrieve command word
                         couture
E08D 0A       ASL        and double it as in-
                         dex for address
E08E AA       TAX
E08F BD C1 D0 LDA D0C1,X Fetch the address (-
                         1) for the command word
E092 48       PHA        Push it onto the
                         stack
E093 BD C0 D0 LDA D0C0,X
E096 48       PHA
E097 60       RTS        and JMP to it via the
                         RTS
E098 A2 09    LDX #09    End up here if no
                         match found for
                         command word
E09A BD C2 E0 LDA E0C2,X and look for a file
                         called ??????.COM to
                         execute
E09D 9D 2B C1 STA C12B,X Set up space for the
                         filename
E0A0 CA       DEX        and .COM extension
E0A1 D0 F7    BNE E09A
E0A3 8E 2B C1 STX C12B
E0A6 20 E8 00 JSR 00E8   Re-fetch the current
                         character
E0A9 9D 2C C1 STA C12C,X Fill in the rest of
                         the filename from text
E0AC E8       INX
E0AD E0 07    CPX #07
E0AF B0 0C    BCS E0BD   Branch to error if
                         name too long
E0B1 20 E2 00 JSR 00E2   Fetch the next
                         character
E0B4 D0 F3    BNE E0A9
E0B6 A9 00    LDA #00
E0B8 85 A9    STA A9
E0BA 4C 06 E1 JMP E106   Jump to load the file
                         ???????.COM
E0BD A2 05    LDX #05    Code for 'Invalid
                         filename' error
E0BF 4C 02 F7 JMP F702   Print error message
                         and exit
```