BASIX

Extension au langage de programmation de l'ORIC

Yann LEGRAND
Club Europe Oric

SOMMAIRE

BASIX, C'EST QUOI ?

BASIX est une extension au langage de programmation de l'ORIC écrite entièrement en langage machine et implantée (de manière native) entre les adresses :

- Oric Atmos : #79B0 et #96FF (activation par CALL#79B0).
- Oric 1 : #78B0 et #96FF (activation par CALL#78B0).

BASIX ajoute plus d'une soixantaine de commandes et de fonctions au Basic, toutes accessibles via les vecteurs '!' et '&'.

La table des mots clefs BASIX se trouve entre les adresses :

- OricAtmos : #7A79 et #7BDA.
- Oric 1 : #7979 et #7AE1

MOT CLEF BASIX	ADRESSE D'EXECUTION	
	Oric Atmos	Oric 1
AND	#89D3	#8969
BOX	#888E	#8793
CAT	#8970	#88B2
CHAR	#83A0	#82CE
CHR$	#8A3E	#89D4
CLI	#7DBF	#7CC7
COL	#8C5B	#8BF3
DATA	#7C42	#7B4A
DEEK	#813E	#806C
DEF	#8571	#8476
DEFCAR	#837B	#82A9
DEG	#87AE	#86B3
DELETE	#8BCC	#8B63
DELREM	#8BCA	#8B61
EDIT	#81E3	#8111
FN	#8A9A	#8A30
FORWARD	#875E	#8663
GOSUB	#85E5	#84EA
GOTO	#8579	#847E
HDEEK	#8143	#8071
HELP	#967E	#967D
HIRES	#7DC1	#7CC9
HPEEK	#8139	#8067
INPUT @	#7E9C	#7E08
INV	#89DB	#8971
LEN	#8877	#877C
LINE	#8D11	#8CA9
LIST	#831E	#824C
LOMEM	#7EDD	#7E0C
LPRINT	#7D1E	#7C26
MAX	#8CED	#8558
MERGE	#88DB	#87E1
MIN	#8CB5	#8C4D
MODIFY	#7F6B	#7E9A
MOVE	#8D8E	#8D26

INSTRUCTION	ADRESSE D'EXECUTION	
	Oric Atmos	Oric 1
OLD	#7F02	#7E31
ON	#8605	#850A
OR	#89D7	#896D
PACK	#8652	#8556
PAINT	#81E9	#8117
PEEK	#8133	#8061
PRINT @	#7E74	#7D7C
POKE	#8A51	#89E7
POP	#933A	#932D
RANDOM	#8123	#8051
READY	#7CFE	#7C06
REM	#7BFA	#7B02
RESTORE	#7BDA	#7AE2
RETURN	#8B7A	#8B11
ROT	#87BF	#86C4
ROUND	#7F1A	#7E49
SCREEN	#84D4	#8402
SEI	#7F18	#7E47
SET	#8CF1	#8954
SORT	#800A	#7F39
STOP	#7E6C	#7D74
SWAP	#8C7B	#8C13
TRON	#87DB	#86E0
TROFF	#91D3	#9166
UNPACK	#8707	#860C
VAL	#8C01	#8B98
VERIFY	N/A	#888A
VLIST	#8DFF	#8D95
VPUT	#8C46	#8BDE
WHERE	#8650	#8554

Cette table de mots clefs peut être aménagée selon les besoins de chaque utilisateur avec les contraintes suivantes :

- Le noyau du programme BASIX est implanté de manière figée entre les adresses #8EC0 et #96FF. Le noyau englobe les commandes !TROFF, !POP, !HELP et &(N), les routines principales et l'interpréteur de lignes de 254 caractères.
- Le nombre de commandes relogeables est de 62.
- Certaines commandes sont indissociables :
  - PRINT @ / INPUT @
  - LOMEM / OLD
  - PEEK / HPEEK / DEEK / HDEEK
  - DEF / GOTO / GOSUB /ON
  - WHERE / PACK / UNPACK
  - FORWARD / DEG / ROT
  - AND / OR / INV
  - FN / RETURN
  - DELREM / DELETE
  - MIN / MAX

L'analyse de la syntaxe est réalisée par une routine implantée entre #7A28 et #7A78(Oric Atmos).

```
7A28  A9 79     LDA #79         ; le pointeur dans la table des mots clefs
7A2A  A0 7A     LDY #7A         ; (début en #7A79) se trouve en
7A2C  85 71     STA 71          ; #71
7A2E  84 72     STY 72          ; et #72
7A30  A0 00     LDY #00
7A32  84 18     STY 18          ; sert au travail d'encodage / décodage des
                                ; mots clefs
7A34  B1 71     LDA (71),Y      ; prendre la longueur du mot clef à analyser
7A36  F0 3E     BEQ 7A76        ; longueur nulle ? Alors poursuivre en #0467
7A38  85 70     STA 70          ; sauver longueur mot clef en #70
7A3A  C8        INY
7A3B  B1 71     LDA (71),Y      ; prendre caractère mot clef
7A3D  88        DEY
7A3E  D1 E9     CMP (E9),Y      ; et comparer avec valeur dans le tampon
                                ; d'entrée
7A40  D0 07     BNE 7A49
7A42  C8        INY
7A43  C4 70     CPY 70          ; a-t-on comparé tous les caractères du mot
                                ; clef ?
7A45  F0 15     BEQ 7A5C
7A47  D0 F1     BNE 7A3A        ; non alors poursuivre la comparaison
7A49  C8        INY
7A4A  C4 70     CPY 70
7A4C  D0 FB     BNE 7A49
7A4E  C8        INY
7A4F  C8        INY
7A50  98        TYA
7A51  38        SEC             ;incrémenter #71-#72 pour pointer sur
                                ; prochain mot clef
7A52  65 71     ADC 71
7A54  85 71     STA 71
7A56  90 D8     BCC 7A30
7A58  E6 72     INC 72
7A5A  D0 D4     BNE 7A30
7A5C  98        TYA             ; mot clef trouvé
7A5D  18        CLC
7A5E  65 E9     ADC E9
7A60  85 E9     STA E9          ; incrémenter TXTPTR du nombre de caractères du mot
clef
7A62  90 02     BCC 7A66
7A64  E6 EA     INC EA
7A66  C8        INY
7A67  B1 71     LDA (71),Y
7A69  85 33     STA 33          ; prendre adresse d'exécution du mot clef
7A6B  C8        INY             ; et stocker en #33-#34
7A6C  B1 71     LDA (71),Y
7A6E  85 34     STA 34
7A70  20 E8 00  JSR 00E8        ; TXTPTR pointe sur '' ou sur ':'
7A73  6C 33 00  JMP (0033)      ; exécuter la commande BASIX
7A76  4C 67 04  JMP 0467        ; suite du traitement du vecteur '!' sous
                                ; SEDORIC
```

Lors de l'initialisation, le programme réalise les modifications suivantes (Oric Atmos) :

- Passe en PAPER 0 : INK 3 et efface l'écran
- Ecrit « BASIX V1.0 Par Ph.NAVEZ&Em.ROSSI » sur la ligne de statut
- Fait un HIMEM#7A28 et initialise le plafond des chaînes également en #7A28
- Détourne #1B-#1C en #8EC0 (normalement #CCB0)
- Remplace JSR ECB9 par JMP ECB9 en #EF (routine de lecture caractère)
- DOKE#8F7F,#F8B2 (NMI : Warm Start)
- DOKE#0248,#8F64 (change traitement du NMI)
- DOKE#9621,#9281
- DOKE#02FC,#95EB (nouvelle adresse de traitement de '&')
- DOKE#02F5,#7A28 (nouvelle adresse de traitement de '!')

LES COMMANDES GRAPHIQUES

La puissance de BASIX est de fournir à l'utilisateur passionné un vaste choix de nouvelles commandes graphiques directement exploitables en mode HIRES.

REMPLISSAGE DE SURFACE

L'opération fastidieuse de remplissage de surface en haute résolution est simplifiée grâce aux quelques commandes suivantes.

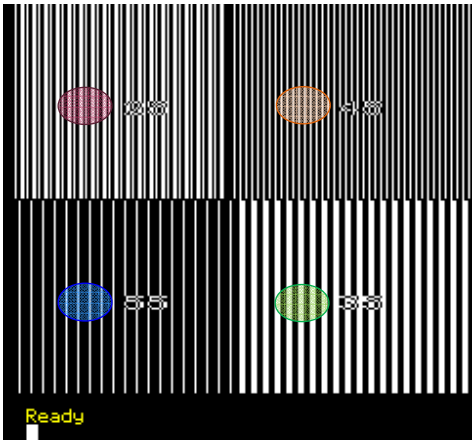
!AND X_1,X_2,Y_1,Y_2,N

Remplit un rectangle qui débute en x_1,y_1 et qui s'achève en x_2,y_2 avec la valeur 'AND n'.

!OR X_1,X_2,Y_1,Y_2,N

Remplit un rectangle qui débute en x_1,y_1 et qui s'achève en x_2,y_2 avec la valeur 'OR n'.

```
10 HIRES
20 !AND 1,39,0,199,64
30!OR1,19,0,100,128+25:!OR19,39,100,199,128+35
40 !OR 19,39,0,100,128+45:!OR 1,19,100,199,128+55
```



!INV X_1,X_2,Y_1,Y_2

Inverse vidéo de la surface délimitée par x_1,y_1 à x_2,y_2.

Attention à l'utilisation de la commande ! INV en mode direct qui provoque un ?ILLEGAL QUANTITY ERROR (et un blocage clavier).

COLORIAGE DE SURFACE

Les deux commandes suivantes permettent un coloriage simplifié de surfaces.

!COL X,N

Place la valeur n dans la colonne x (compris entre 0 et 39). !COL fonctionne en mode TEXT ou en mode HIRES.

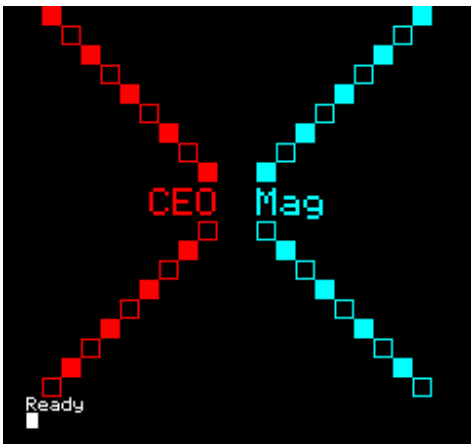
!PAINT X,Y

Remplit une surface à partir des coordonnées x et y.

Petite précaution d'emploi : lors de l'utilisation de la commande PAINT, la surface à remplir doit être parfaitement délimitée, faute de quoi l'intégralité de l'écran HIRES est mise en couleur.

Il faut aussi savoir que la combinaison de touche CTRL+C est inactive lors du fonctionnement de la commande PAINT.

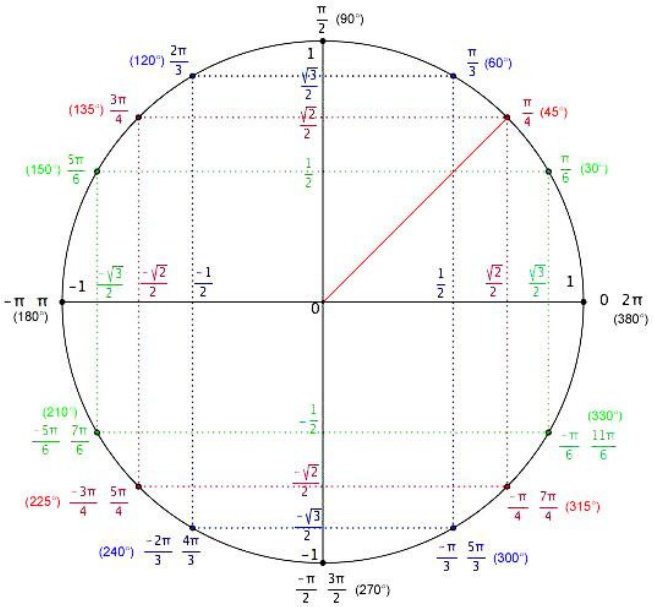
```
10 HIRES:J=190
20 FOR I=0 TO 190 STEP 10
30 IF I=90 OR I=100 THEN 80
40 CURSET I+20,I,0
50 BOX 10,10,1
60 CURSET I+20,J,0
70 BOX 10,10,1
80 J=J-10
90 NEXT:J=195
100 FOR I=25 TO 195 STEP 20
110 IF I=105ORI=125 THEN 140
120 !PAINT I,I-20
130 !PAINT I+10,J-10
140 J=J-20
150 NEXT
160 !PAINT 215,5:!PAINT 205,185
170 !PAINT 105,85:!PAINT 135,85
180 !COL 0,1
190 !COL 19,6:!COL 20,6
200 A$="CEO"
210 !CHAR A$,75,93,2,2,H
220 A$="Mag"
230 !CHAR A$,130,93,2,2,H
```



POSITIONNEMENT ET MOUVEMENT

Petit rappel de trigonométrie

Pour déterminer la localisation d'un point d'arrivé relatif à un point de départ selon un angle α, il faut réaliser une projection sur le cercle trigonométrique.



Dans l'exemple ci-dessus, la projection de l'angle π/4 (45°) donne la même valeur √2/2 pour le sinus et le cosinus.

Si l'on souhaite une droite d'angle 45° et de longueur 10, il suffit donc de :

- multiplier 10 par √2/2 pour obtenir le déplacement relatif en abscisse
- faire de même pour le déplacement relatif en ordonnée

Note : en raison de l'inversion sur l'axe des ordonnées sur Oric, il faut prendre la valeur négative du sinus de l'angle α.

Grâce aux quelques commandes suivantes, il devient désormais aisé de positionner et de bouger le pointeur en haute résolution.

!DEG N

Définit la valeur en degrés qui sera utilisée pour la commande `FORWARD`.

!FORWARD N,FB

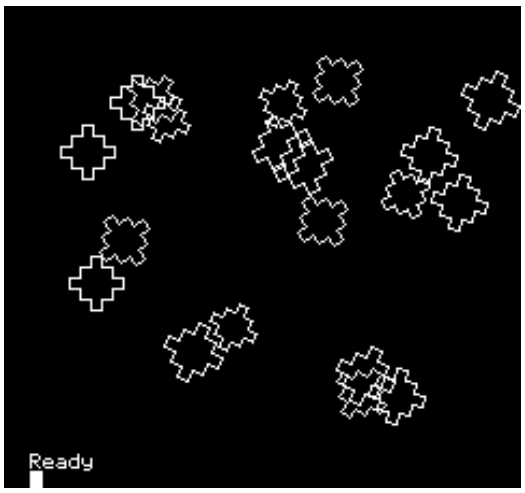
Réalise un tracé de n pixels (depuis la valeur courante) selon le code fb et l'angle définit par `!DEG`.

Equivaut à la commande Sedoric`LINE n,fb`.

!ROT N

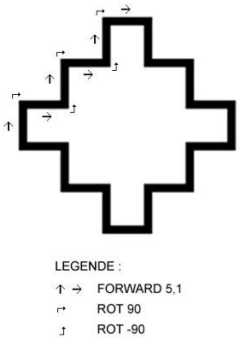
Réalise une rotation de n degrés.

```
10 HIRES
20 FOR N=1 TO 20
30 X=INT(RND(1)*200)+30
40 Y=INT(RND(1)*160)+10
50 D=INT(RND(1)*90)+1
60 IF RND(1)>0.5 THEN D=-D
70 CURSET X,Y,0
80 GOSUB 500
90 NEXT N
100 END
500 !DEG D:GOSUB 1000
510 GOSUB 1010
520 GOSUB 1010
530 GOSUB 1010
540 RETURN
1000 !FORWARD 5,1
1010 !ROT 90
1020 !FORWARD 5,1
1030 !ROT -90
1040 !FORWARD 5,1
1050 !ROT 90
1060 !FORWARD 5,1
1070 !ROT -90
1080 !FORWARD 5,1
1090 !ROT 90
1100 !FORWARD 5,1
1110 RETURN
```



Explications :

- lignes 30 à 60 : tirer aléatoirement abscisse X et ordonnée Y du point de départ pour l'affichage de la forme géométrique selon un angle D également tiré au hasard.
- lignes 500 à 530 : tracer les 4 côtés de la figure géométrique selon un angle D.
- lignes 1000 à 1110 : tracer un côté de la figure géométrique (voir ci-dessous).



COMMANDE CHAR

La commande CHAR est l'une des plus évoluées jamais développée pour le mode haute résolution de l'Oric. Elle permet d'écrire du texte directement dans ce mode.

Syntaxe : **!CHAR A$,X,Y,$n_1$,$n_2$,H ou V**

A$: chaîne alphanumérique d'une longueur maximale de 40 caractères

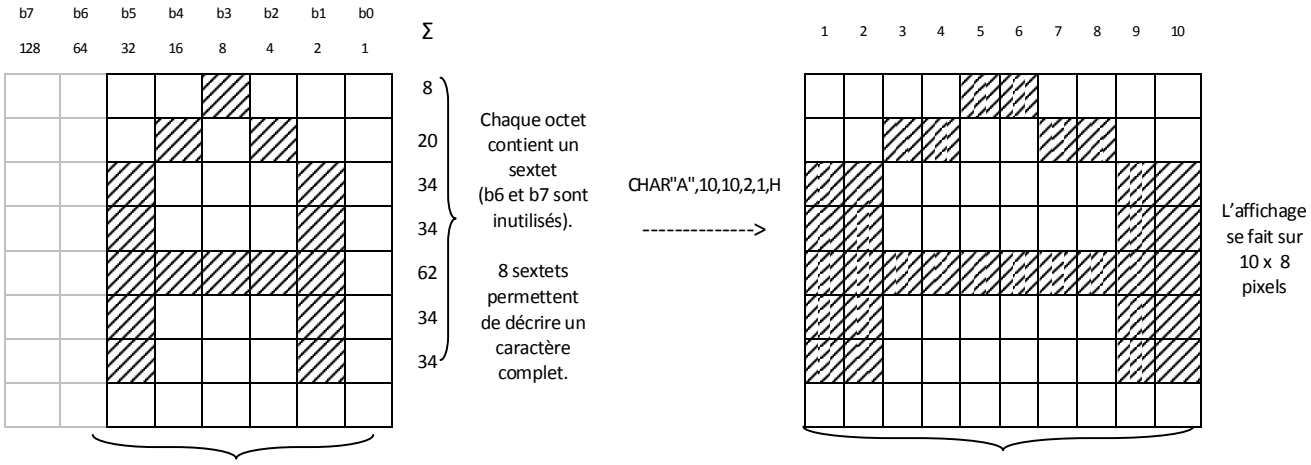
X,Y : coordonnées auxquelles doit être affichée A$

n_1 : facteur d'agrandissement horizontal (max. = 30)

n_2 : facteur d'agrandissement vertical (max. = 25)

H ou V : affichage 'H'orizontal ou 'V'ertical

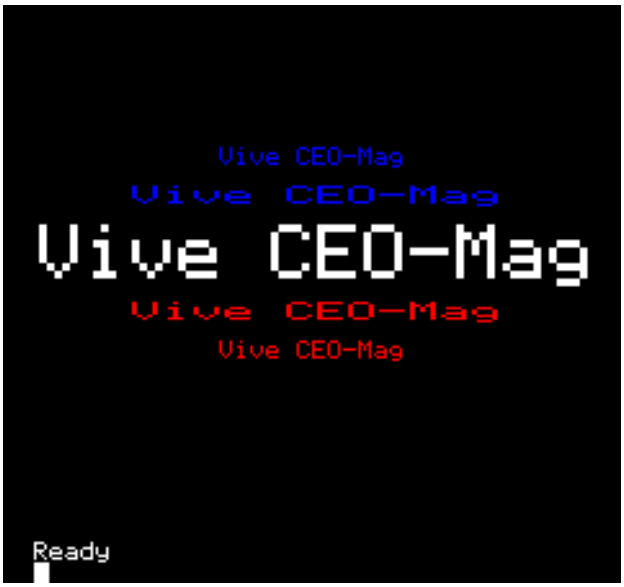
Exemple de traitement de la commande CHAR (facteur d'agrandissement horizontal $n_1 = 2$) :



Quelques précisions :

- En raison de l'appel de la sous-routine en 00E2, l'utilisation de la commande `CHAR` en mode direct est possible mais génère systématiquement un `?SYNTAX ERROR`.
- L'interdiction des interruptions empêche l'arrêt de la commande `CHAR` par la combinaison de touche CTRL + 'C'.
- Le repositionnement du curseur par `CURMOV` en fin de routine ne permet pas d'utiliser la commande Sedoric `HCUR`.
- En cas dépassement de fin d'écran, le curseur est repositionné selon les coordonnées initiales (avec poursuite de l'affichage).

```
10 REM Demo commande CHAR
20 HIRES
30 FOR I=1 TO 2
40 Y=40+(I*15)
50 READ X
60 !CHAR"Vive CEO-Mag",X,Y,I,1,H
70 NEXT
80 FOR I=2 TO 1 STEP-1
90 Y=85+((4-I)*15)
100 READ X
110 !CHAR"Vive CEO-Mag",X,Y,I,1,H
120 NEXT
130 !CHAR"Vive CEO-Mag",14,85,3,3,H
140 CURSET 0,55,0:FILL 30,1,4
150 CURSET 0,115,0:FILL 30,1,1
200 DATA 84,50
210 DATA 50,84
```



COMPACTAGE ET DECOMPACTAGE D'ECRAN HAUTE RESOLUTION

BASIX comprend trois commandes particulièrement puissantes qui permettent de compacter et de décompacter des écrans haute résolution, autorisant ainsi de courtes animations.

Quelques précisions :

- Comme pour la majorité des instructions graphiques utilisables en BasiX, l'interdiction des interruptions empêche l'arrêt des commandes !PACK / !UNPACK par la combinaison de touche CTRL + 'C'.
- En mode direct, la commande !WHERE se termine par un blocage du clavier.
- Une mauvaise adresse provoque l'apparition du message ?DISP TYPE MISMATCH ERRORet le blocage du clavier.
- La commande !UNPACK récupère l'intégralité de l'écran HIRES compacté ; aucune vidange préalable n'est donc nécessaire.
- Il y a réellement un compactage de l'écran HIRES en mémoire. Les essais réalisés avec le programme de démonstration montrent en effet que l'écran HIRES, qui occupe 7999 octets, peut être compacté jusqu'à 5,6 fois (le compactage est fonction de la complexité de l'image haute résolution à traiter). D'ailleurs, les 5 écrans HIRES compactés en mémoire dépassent rarement l'adresse #3500, ce qui prouve bien la grande efficience du compactage (environ 4 fois).

!PACK AD,AF

AD : adresse à laquelle l'écran HIRES sera compacté.

AF : adresse du dernier octet de l'écran HIRES compacté.

!UNPACK AD

AD : adresse du premier octet de l'écran HIRES compacté

!WHERE AD,AF

Fonctionne de manière identique à !PACK mais sans stockage. Cette commande présente l'avantage de déterminer quelle serait l'adresse de fin de l'écran compacté et de s'assurer ainsi que l'on ne risque pas d'écraser la zone de stockage de BASIX (**#7A28 - #96FF**).

Le programme de démonstration qui suit réalise l'affichage des chiffres 1 à 5 à l'écran (via la commande !CHAR) et compacte chaque écran à partir de l'adresse #1000 (lignes 15 à 110).

Le programme récupère ensuite les écrans compactés dans l'ordre inverse de leur sauvegarde (lignes 120 à 150).

La routine implantée en #7A00 permet de vidanger l'écran HIRES par un simple CALL#7A00.

```
10 GOSUB 500
15 HIRES:AD=#FFF
20 FOR I=1 TO 5
30 A$=STR$(I):A$=RIGHT$(A$,1)
40 N=INT(RND(1)*20)+10
50 M=INT(RND(1)*15)+10
60 X=N+20:Y=M+5
70 J=INT(RND(1)*7)+1
80 INK J
90 !CHAR A$,X,Y,N,M,H
100 GOSUB 200:CALL#7A00
110 NEXT I
120 FOR I=4 TO 1 STEP -1
130 GOSUB 300:WAIT 50
140 NEXT I
150 !UNPACK #1000
160 END
200 REM Compactageecran
210 !PACK AD+1,AD
220 T(I)=AD
230 RETURN
300 REM Decompactageecran
310 !UNPACK T(I)+1
320 RETURN
490 REM Programme L/M vidageecran
500A=#7A00:F=#7A1E:L=100:REPEAT:FORA=A TO A+15:READ C$
510 K=VAL("#"+C$):S=S+K+65536*(S+K> 65535):IF A<=F THEN POKE A,K
520 NEXT:READ D$:IF S=VAL("#"+D$) THEN L=L+5:UNTIL A>F:END
530 PING:PRINT"Erreurligne";L
540DATAA2,A0,A0,00,8C,0D,7A,8E,0E,7A,A9,40,8D,00,BF,EE,072E
550DATA0D,7A,D0,F8,EE,0E,7A,AD,0E,7A,C9,BF,D0,EC,60,00,0FCC
```

DESSINS EN HAUTE RESOLUTION

Les deux commandes suivantes, quoique déjà existantes sous SEDORIC, permettent de tracer des dessins très facilement en mode haute résolution.

!BOX N_1,N_2,FB

Trace un rectangle de n_1 pixels sur n_2 pixels avec le code fb.

Le point de départ du tracé est le coin supérieur gauche du rectangle.

```
10 HIRES
20 !CHAR "Demo commande BOX",18,0,2,2,H
30 N1=5:N2=5
40 X1=114:Y1=104
50 FORI=1TO18
60 CURSET 119,109,0
70 !BOX N1,N2,1
80 CURSET X1,Y1,0
90 !BOX N1,N2,1
100 N1=N1+5:N2=N1
110 X1=X1-5:Y1=Y1-5
120 NEXT I
130 N1=90:N2=5
140 X1=29:Y2=104
150 FORI=1TO18
160 CURSET X1,109,0
170 !BOX N1,N1,1
180 CURSET 119,Y2,0
190 !BOX N2,N2,1
200 X1=X1+5:N1=N1-5
210 Y2=Y2-5:N2=N2+5
220 NEXT I
230 X1=30:Y1=20
240 X2=203:Y2=25
250 X3=35:Y3=193
260 X4=208:Y4=198
270 FORI=1TO9
280 !PAINT X1,Y1
290 !PAINT X2,Y2
300 !PAINT X3,Y3
310 !PAINT X4,Y4
320 X1=X1+10:Y1=Y1+10
330 X2=X2-10:Y2=Y2+10
340 X3=X3+10:Y3=Y3-10
350 X4=X4-10:Y4=Y4-10
360 NEXT I
370 FORI=1TO10
380 N=INT(RND(1)*4)+1
390 ON N GOSUB 500,510,520,530
400 WAIT 50
410 NEXT I
499 END
500 !INV5,19,19,109:RETURN
510 !INV20,34,109,199:RETURN
520 !INV5,19,109,199:RETURN
530 !INV20,34,19,109:RETURN
```

!LINE FB,(X1,Y1 TO X2,Y2)

Trace une ligne des coordonnées x_1,y_1 aux coordonnées x_2,y_2 avec le code fb.

```
10 HIRES:GOSUB 500
20 X=6:Y=100
30 FORI=1TO20
40 !LINE 1,(0,Y TO X,0)
50 X=X+6
60 Y=Y-5
70 NEXT
80 X=120
90 FORI=1TO20
100 !LINE 1,(X,0 TO 239,Y)
110 X=X+6
120 Y=Y+5
130 NEXT
140 X=239
150 FORI=1TO20
160 !LINE 1,(239,Y TO X,199)
170 X=X-6
180 Y=Y+5
190 NEXT
200 Y=199
210 FORI=1TO20
220 !LINE 1,(X,199 TO 0,Y)
230 X=X-6
240 Y=Y-5
250 NEXT
260 !CHAR "Vive le",58,65,3,3,H
270 !CHAR "CEO-Mag",38,115,4,4,H
499 END
500 !LINE 1,(0,0 TO 239,0)
510 !LINE 1,(239,0 TO 239,199)
520 !LINE 1,(239,199 TO 0,199)
530 !LINE 1,(0,199 TO 0,0)
540 CURSET 210,65,0:FILL 82,1,7
550 CURSET 50,65,0:FILL 21,1,4
560 CURSET 30,115,0:FILL 32,1,1
570 RETURN
```

COMMANDES AUTRES

!HIRES

Lance l'impression de l'écran HIRES. L'impression n'est possible que sur une véritable imprimante, le code émis n'étant pas exploitable dans un fichier "printer.txt" (créé en temps normal sous Euphoric lors d'une commande LPRINT).

!SCREEN N_1,N_2,N_3,N_4

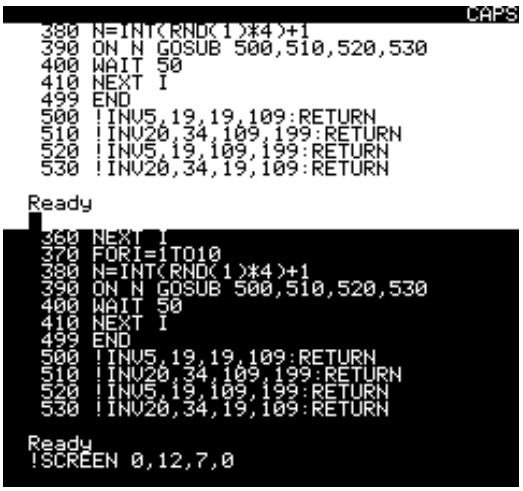
Créé un sous-écran texte de n_2 lignes à partir de la ligne n_1, avec les couleurs `PAPER` n_3 et `INK` n_4.

$$0 < n_1,n_2 < 27 \text{ et } 0 < (n_1+n_2) < 27$$

n_3 et n_4 sont facultatifs

Si n_1 et n_2 sont omis, `!SCREEN` équivaut à un simple `CLS` (avec retour à un écran texte classique).

Lors du passage des paramètres n_3 et n_4, le sous-écran texte créé est bien dans une couleur différente du reste de l'écran (voir exemple ci-dessous).



LE TRAITEMENT DES VARIABLES

BASIX offre de nouvelles commandes intéressantes pour faciliter le traitement des variables.

!SORT TABLEAU(0)

Tri le tableau de la plus petite à la plus grande valeur.

Le tableau peut être de différents types :

- Numérique : réel (A(10)) ou entier (A%(10))
- Alphanumérique (A$(10))

Notes :

- Après le tri, la valeur la plus élevée du tableau se retrouve stockée au rang n+1 et le chiffre zéro est stocké au rang 0.
- Il faut obligatoirement saisir '(0)' à la suite de la variable tableau, faute de quoi il s'ensuit un message du type `?TYPE MISMATCH ERROR` et un blocage clavier.

```
10 REM Demo fonction SORT
20 RELEASE:CLS
30 GOSUB 1010
40 PRINT:PRINTSPC(8)"TABLEAU DE SCORES:":
PRINT:PRINT"Avant classement"
50 FORI=0TO9
60 PLOT2,I+6,STR$(X(I))+"..."+A$(I)
70 NEXTI
80 !SORT X(0)
90 FORI=0TO10
100 FORJ=0TO10
110 IFX(I)=Y(J)THENB$(I)=A$(J)
120 NEXTJ,I
490 PLOT20,3,"Apres classement":J=6
500 FORI=10TO1STEP-1
510 PLOT20,J,STR$(X(I))+"..."+B$(I)
520 J=J+1
530 NEXTI
540 PRINT@8,18;"Fin demo fonction SORT"
550 END
1000 RANDOM
1010 FORI=0TO9
1020 READ A$(I)
1030 X(I)=INT(RND(1)*10000)+500
1040 Y(I)=X(I)
1050 NEXTI
1060 RETURN
2000 DATA THIBAUT,AMANDINE,SAMSON,MARTHE
2010 DATA ABEL,CLARISSE,AUGUSTIN,INGRID
2020 DATA ARISTIDE,INES
```

!MIN MI=A,B,C,D,…

Détermine la valeur minimum d'une suite de nombres et l'attribue à la variable MI.

!MAX MA=A,B,C,D,…

Détermine la valeur maximum d'une suite de nombres et l'attribue à la variable MA.

```
10 REM INSTRUCTIONS !MIN ET !MAX
20 REM
30 TEXT:CLS:PAPER0:INK2:POKE618,2
40 N=30:DIMB%(N):D2$="DEMONSTRATION
DESINSTRUCTIONS"
50 FORI=0TON:B%(I)=INT(RND(1)*2000)+1:NEXT:
M1=2000:M2=0
60 FORI=0TON:!MIN M1=M1,B%(I):!MAX
M2=M2,B%(I):NEXT
70 RO$=CHR$(129):JA$=CHR$(131):MA$=CHR$(133)
80 PRINTJA$D2$:PRINTRO$"      !MAX ET !MIN"
90 PRINT:PRINT:PRINTMA$"Voici une liste d'entiers
aleatoire :":PRINT:PRINT:
PRINT:FORI=0TON:PRINTB%(I),:NEXT:PRINT:PRINT:PRIN
TRO$"Minimum de la liste :
 ",M1:PRINT:PRINTRO$"Maximum de la liste : ",M2
100 PRINT:PRINT"Vous pouvez verifier !"
110 POKE618,7
```

!ROUND VAR,EXPRESSION

Arrondit la variable 'var' d'un nombre équivalent au résultat de 'expression', 'var' est de type réel.
La formule de troncature utilisée par la commande ROUND est la suivante :

$$\frac{\text{INT}\,[(10^{n}.x) + 0.5]}{10^{n}}$$

Avec n le nombre de chiffres après la virgule et x le nombre à arrondir.

!SET VAL TO VAR_1,VAR_2,VAR_3,…VAR_I

Initialise les variables var_1 à var_i avec la valeur val (numérique exclusivement).
Cette commande est intéressante lorsqu'il faut réinitialiser des variables après une partie.

!SWAP VAR_1,VAR_2

Intervertit les contenus respectifs des deux variables var_1 et var_2 de même type (dans le cas contraire, vous avez droit à un ?TYPE MISMATCH ERROR).
Pour des variables alphanumériques, il est impératif de respecter la même longueur de chaîne.

!VAL VAR = A$

Transfère dans la variable 'var' le résultat de l'expression numérique contenu dans A$ (longueur maximum de 79 caractères).
Cette commande est d'un intérêt limité, puisqu'elle réalise la même opération qu'une affectation classique de résultat à une variable.

LES COMMANDES D'AIDE A LA PROGRAMMATION

Sous BASIX, les commandes d'aide à la programmation sont particulièrement nombreuses.

Certaines d'entre-elles sont d'ailleurs assez similaires à des commandes SEDORIC connues.

!DELETE N_1,N_2

Détruit les lignes du programme Basic comprises entre n_1 et n_2 (incluses).

!DELREM N_1,N_2

Détruit les lignes du programme Basic comportant des REM, de la ligne n_1 à la ligne n_2.

!DEF "ETIQUETTE"

Permet de placer une "étiquette" en début de ligne Basic.

!EDIT N

Edite la ligne Basic n (ce qui en réalité revient au même que la commande Basic EDIT…).

!GOSUB "ETIQUETTE"

Se rend à la ligne comportant l'"étiquette" définit précédemment par la commande !DEF.

!GOTO "ETIQUETTE"

Idem !GOSUB"étiquette".

!HELP

Affiche la liste des commandes BASIX actuellement exploitables.

Le listing peut être mis en pause à tout moment par appui sur la touche `espace´ puis stoppé par la combinaison CTRL+C.

!INPUT @X,Y;"MESSAGE";VAR

Positionne la commande INPUT aux coordonnées x,y.

!LEN N

Définit la longueur maximale d'une ligne Basic (0 < n < 255).

Petite particularité : le paramètre n peut être une variable décimale.

!LIST EXPRESSION

Recherche et liste l'expression demandée.

Cette commande s'apparente fortement à la commande SEEK sous SEDORIC à la différence près que le caractère joker '£' n'est pas pris en charge.

!MERGE "NOM PROGRAMME"

Fusionne le programme demandé (au format TAP) avec celui en mémoire.Lors de la fusion, le message "Working .." s'affiche sur la ligne de statut.

Cette commande est plus puissante que l'option ´J` des commandes CLOAD ou LOAD, car il y a une vraie tentative de fusion du programme avec un respect des différents numéros de ligne Basic.

!MODIFY EXPRESSION$_1$,EXPRESSION$_2$(,LIGNE$_1$,LIGNE$_2$)

Réalise le remplacement de l'expression 1 par l'expression 2 dans le programme Basic, de la ligne 1 à la ligne 2 (facultatif).

L'emploi de la ligne Basic de départ de la recherche doit impérativement être suivi de la ligne Basic de fin de recherche, sinon cela déclenche un ?SYNTAX ERROR.

Comme pour son homologue CHANGE sous SEDORIC, cette commande n'est exploitable qu'en mode direct.

Son utilisation dans un programme reste possible mais provoque la sortie dudit programme après exécution.

!OLD

Récupère un programme effacé par NEW.

!ON I GOTO EXP$_1$,EXP$_2$,…,EXP$_I$

Se rend à ligne BASIC contenant l'expressionexp$_i$.

L'expression exp$_i$ peut être numérique ou alphanumérique. Dans ce dernier cas, elle devra être déclarée par la commande !DEF.

!ON I GOSUB EXP$_1$,EXP$_2$,…,EXP$_I$

Se rend au sous-programme situé en ligne exp$_i$.

```
10 CLS:!GOSUB"INIT"
20 INPUT"NUMERO SOUS PROGRAMME ";N:PRINT
30 !ON N GOSUB"SP1","SP2","SP3"
40 END
100 !DEF"SP1"
110 PRINTM$(1)
120 RETURN
130 !DEF"SP2"
140 PRINTM$(2)
150 RETURN
160 !DEF"SP3"
170 PRINTM$(3)
180 RETURN
200 !DEF"INIT"
210 FORI=1TO3
220 M$(I)="BIENVENUE DANS LE S/P"+STR$(I)
230 NEXTI
240 RETURN
```

!RANDOM

Initialise le générateur de nombres aléatoires au hasard.

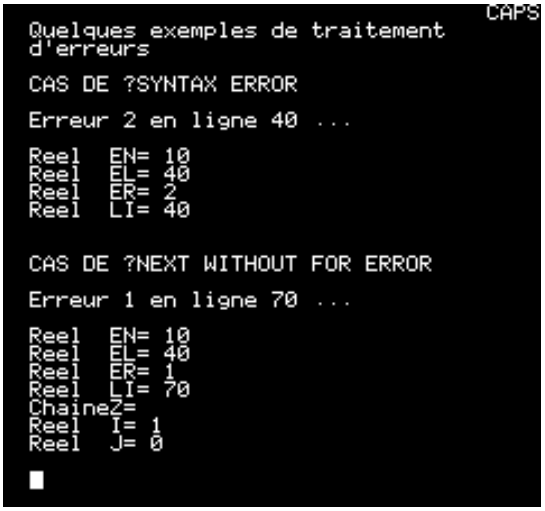
!RESTORE N

Place le pointeur des DATA à la ligne n.

!TRON N

Active le gestionnaire d'erreurs BASIC. Le traitement de l'erreur se fait à la ligne n.

```
10 TEXT:CLS:!TRON 1000:DIM N(20)
20 PRINT"Quelques exemples de traitement":PRINT"d'erreurs":PRINT
30 PRINT"CAS DE ?SYNTAX ERROR"
40 PRNT
50 PRINT"CAS DE ?NEXT WITHOUT FOR ERROR"
60 FOR I=1 TO 10
70 NEXT J
80 PRINT"CAS DE ?REDIM'D ARRAY ERROR"
90 DIM N(30)
100 PRINT"CAS DE ?DISP TYPE MISMATCH ERROR"
110 CURSET 0,0,0
120 PRINT"CAS DE ?RETURN WITHOUT GOSUB ERROR"
130 RETURN
140 PRINT"CAS DE ?BAD UNTIL ERROR"
150 UNTIL A=10
160 !TROFF:END
1000 PRINT"Erreur";ER;"enligne";LI;"..."
1010 !VLIST
1020 POKE#2DF,0:GET Z$
1030 PRINT
1040 GOTO LI+10
```



!TROFF

Désactive le gestionnaire d'erreurs BASIC.

Petite particularité : il faut impérativement désactiver le gestionnaire d'erreurs en fin de programme, faute de quoi le gestionnaire boucle sur lui-même jusqu'à un RESET de la machine.

!VLIST

Liste les variables en cours d'utilisation au moment du traitement d'une erreur.

L'appui sur la touche 'SPC' provoque une pause dans le listing.

LES COMMANDES DE MANIPULATION DE LA MEMOIRE

BASIX permet d'agir sur la mémoire grâce à quelques commandes relativement simples d'emploi.

!DEEK ADR

Affiche les valeurs stockées en mémoire, à partir des deux emplacements adr et adr+1 et ce jusqu'à appui sur la combinaison de touches CTRL + 'C'.

L'appui sur 'espace' permet de stopper momentanément l'affichage des valeurs à l'écran.

!HDEEK ADR

Fonctionne de la même manière que !DEEKadr avec des valeurs hexadécimales.

!PEEK ADR

Fonctionne de la même manière que !DEEKadr, mais à partir du seul emplacement adr.

!HPEEK ADR

Fonctionne de la même manière que !PEEKadr avec des valeurs hexadécimales.

Cette commande s'apparente fortement à un DUMP mémoire.

Petite singularité : les commandes !DEEK, !HDEEK, !PEEK et !HPEEK ne sont pas regardantes sur les caractères qui suivent l'adresse adr. En d'autres termes, la commande !HPEEK#79B0,A$ ne provoque aucune erreur de la part de l'interpréteur.

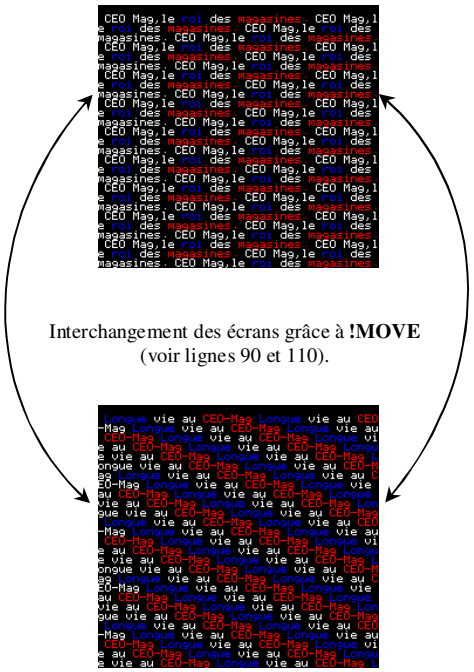
!POKE ADR_1,ADR_2,N_1,N_2,N_3,N_I,…

Place les valeurs n_1 à n_i à partir de l'adresse adr_1, puis répète la séquence n_1 à n_i jusqu'à l'adresse adr_2.

!MOVE ADR_1,ADR_2,N

Copie la zone mémoire (adr_1+n) à partir de l'adresse adr_2.

```
10 TEXT:CLS:PAPER0:INK7:POKE48035,0
20 !POKE#BBA8,#BFDF,07,67,69,79,32,77,
97,103,44,108,101,04,114,111,105,07,100
,101,115,01,109,97,103,97,115,105,110,101,115,46
30 WAIT 100
40 !MOVE#BBA8,#3000,1080
50 !POKE#BBA8,#BFDF,04,76,111,110,103,
117,101,07,118,105,101,32,97,117,01,67,69,79,45,77,97,103
60 WAIT 100
70 !MOVE#BBA8,#3500,1080
80 REPEAT
90 !MOVE#3000,#BBA8,1080
100 WAIT 100
110 !MOVE#3500,#BBA8,1080
120 WAIT 100
130 UNTIL PEEK(#208)=#A9
140 POKE48035,7
```



Interchangement des écrans grâce à **!MOVE** (voir lignes 90 et 110).

!LOMEM ADR

Fixe le début des programmes Basic à l'adresse adr.

Une adresse supérieure à celle fixée par la commande Basic HIMEM provoque un ?ILLEGAL QUANTITY ERROR.

!DEFCAR $X_1,N_1,N_2,N_3,N_4,N_5,N_6,N_7,N_8;X_2,..N_I$

Sauve les valeurs n_1 à n_8 à l'emplacement #B3FF+x_i*8 (mode TEXT). X_i compris entre 0 et 9.

La commande !DEFCAR présente l'inconvénient de ne fonctionner que pour la redéfinition des caractères '0' à '9'.

Une autre programmation aurait permis de redéfinir tout type de caractère. Il aurait suffi pour cela de faire appel à la sous-routine D8C8 (prendre un entier dans le registre X) dès le début de l'interprétation de la commande.

!VPUT EXPR TO ADR

Stocke le résultat de l'expression numérique 'expr' à l'adresse 'adr'.

Cette commande ne présente pas grand intérêt, hormis pour les développeurs en langage machine qui souhaitent savoir comment est codé un nombre réel en calcul flottant.

FONCTIONS RECURSIVES

Les trois commandes **!FN**, **!POP** et **!RETURN** permettent de définir des fonctions dites récursives.

N'étant pas un excellent mathématicien, je ne suis pas en mesure d'en développer l'utilisation.

Cependant, le programme de démonstration de BasiX exploite les possibilités de ces commandes pour solutionner le problème des "Tours de Hanoï".

Le problème des tours de Hanoï est un problème mathématique célèbre inventé par le mathématicien français Édouard LUCAS en 1883.

Le problème, qui peut devenir très vite complexe, peut être décrit simplement.[1]

On dispose de trois "tours" A (départ), B (transit) et C (arrivée) formées d'un empilement plus ou moins grand de disques de telle sorte que chaque disque ait un diamètre inférieur à son prédécesseur. Ainsi, le plus grand disque de chaque tour se situe à leur base et le plus petit sur leur sommet.

Au départ, tous les disques (supposons qu'il y en ait n) se trouvent empilés suivant les règles précédemment décrites sur la tour A.

L'objectif est de déplacer tous les disques de la tour A vers la tour C en s'aidant uniquement de la tour B, tout en respectant les règles suivantes :

- On ne peut bouger qu'un seul disque par étape et cela doit toujours être le plus petit (celui qui se trouve au sommet) ;
- À chaque étape, la règle d'organisation des tours décrite précédemment doit être respectée.

Pour résoudre le problème des tours de Hanoï, il faut procéder de manière récursive.

La logique est la suivante : "isoler le plus grand disque de A puis le placer à la base de la tour de destination C puis appliquer le même algorithme de B vers C".

En résumé, l'algorithme de résolution des tours de Hanoï sur un nombre n de disques est donc :

- Déplacer n disques de A vers C en passant par B :
  - Déplacer (n-1) disques de A vers B en passant par C;
  - Déplacer 1 disque de A vers C ;
  - Déplacer (n-1) disques de B vers C en passant par A.

L'évaluation de la complexité de cet algorithme est la suivante. Il faut chercher à savoir combien de déplacements de disques sont nécessaires pour arriver à nos fins. La fonction calculant ceci peut être définie ainsi :

$$f(1) = 1; f(n) = 2*f(n-1)+1$$

On peut démontrer par récurrence que ceci est égal à f(n) = 2^n-1. On a donc f(n+1) = 2*f(n)+1. En supposant que f(n) = 2^n-1, on a f(n+1) = 2*(2^n-1)+1.

On développe ensuite en f(n+1) = 2*2^n-2+1 ce qui nous donne au final f(n+1) = 2^(n+1)-1.

On a ainsi démontré que si c'est vrai pour n, alors c'est vrai pour (n+1). Comme on a f(1) = 2*0+1 = 2^1-1 = 1, c'est vrai pour tous les n supérieurs à 1 et 1.

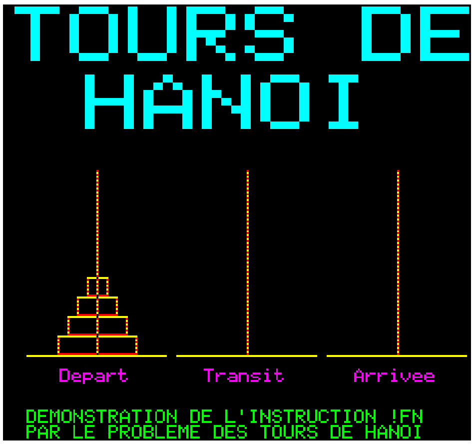
La complexité exacte de l'algorithme présenté ci-dessus est donc en 2^n-1 pour n disques à déplacer de la tour A vers la tour C.

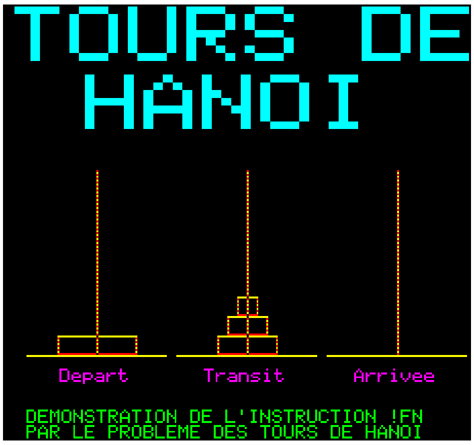
[1] http://www.siteduzero.com/tutoriel-3-126911-les-tours-de-hanoi.html

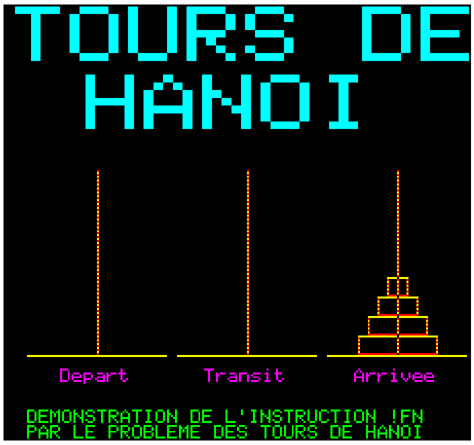
Voici ce que donne la résolution de ce problème en BasiX (programme de démo livré avec BasiX).

```
350 REM
360 REM       TOURS DE HANOI
370 REM
380 CLEAR:DIMT(4):!SET #2E00 TO T(0),T(1),T(2),T(3),T(4):GOSUB2170:PO=1:TEXT
390 CLS:PAPER0:INK2:PRINT:PRINT:
PRINTCHR$(129)"TOURS DE HANOI":PRINT:PRINT
400 !PRINT @2,6;"Nombre de disques a deplacer (1 a 7)?":GOSUB2150:IFA$=""THEN
N=3:GOTO420
410 N=VAL(A$):IFN<1ORN>7THEN!CHR$ 12,7
:GOTO390
420 DIM P(2,N):HIRES:POKE 618,10
430 CLS:PRINTD1$"!CHAR"
440 FOR X=1 TO N:P(0,X)=N-X+1:NEXT
450 FILL66,1,6:FORI=1TO60:FILL1,1,3
:FILL1,1,1:NEXT:!CHAR"TOURS DE",4,1,5,4,H
:!CHAR"HANOI",42,36,5,4,H
460 ZX=187:CURSET0,ZX,0:FORI=1TO10:
FILL1,1,5:NEXT:!CHAR"Depart",29,ZX,1,1,H:!CHAR"Transit",103,ZX,1,1,H:!CHAR"Arrivee",180,ZX
,1,1,H
470 P(0,0)=N:FOR X=0 TO 2
480 !LINE 1,(12+X*77,180 TO 3+X*77,180),
1,(48+X*77,85 TO 48+X*77,179)
490 NEXT:!SET 0 TO DEPL,MA
500 FOR X=1 TO N
510 CURSET 48-P(0,X)*5,180-X*10,3
520 !BOX P(0,X)*10,9,2
530 NEXT:P=&(10)P(0,0)
540 PRINT:PRINTD1$"!FN":PRINT"PAR LE PROBLEME DES TOURS DE HANOI";
550 REM RESOLUTION DU PROBLEME
560 REM   EN UNE INSTRUCTION!
570 !FN (610,(DEPL,MA),N,0,2,1,DEPL,P,0)
580 FOR X=1 TO N:!PAINT 200,185-X*10:
!PAINT 204,185-X*10:NEXT:!LINE 1,(202,85 TO 202,179)
590 CLS:PRINT"Tapez 'ESC' pour recommencer, une":PRINT"autre touche pour continuer la
demo":!POP 0
600 A$=" ":GOSUB2150:IF A$=CHR$(270THEN380
ELSE!GOSUB"COMPECRAN":!GOTO"DEMOIMP"
610 DEF N,X,Y,Z,DEPL,P,MA
```

```
620 IF N>0 THEN !FN (610,(DEPL,MA),N-1,X,Z,Y,DEPL,P,MA):GOSUB 640:!FN (610,(DEPL,MA),N-
1,Z,Y,X,DEPL,P,MA):
!RETURN DEPL+1,MA
630 !MAX MA=MA,&(9):!RETURN DEPL,MA
640 REM DEPLACEMENT DE X VERS Y
650 DD=P+5*X+(&(13)P+5*X)*15:
DF=P+5*Y+(&(13)P+5*Y)*15:TA=&(13)DD:
HA=&(13)P+X*5:VI=180-HA*10
660 FOR A=VI TO 85 STEP -10
670 CURSET 48+X*77-TA*5,A,3:!BOX TA*10,9,2
:CURMOV 0,-10,3:!BOX TA*10,9,2:NEXT
680 !BOX TA*10,9,2:CURMOV 77*(Y-X),0,3:
!BOX TA*10,9,2:HA=1+&(13)P+5*Y:VF=180-HA*10
690 FOR A=85 TO VF STEP 10
700 !BOX TA*10,9,2:CURMOV 0,10,3:!BOX TA*10,9,2:NEXT
710 !VPUT 0 TO DD
720 DD=&(13)P+5*X:DD=DD-1:!VPUT DD TO P+5*X
730 !VPUT TA TO DF+15
740 DD=&(13)P+5*Y:DD=DD+1:!VPUT DD TO P+5*Y
750 RETURN
```







COMMANDES DIVERSES

!CAT "NOM PROGRAMME"

Affiche les caractéristiques liées à l'en-tête de "nom programme" (début, fin, AUTO, Basic / LM, nom programme).

!CHR$ $N_1, N_2, N_3, \ldots N_I$

Envoi à l'écran les caractères n_1 à n_i.

Avec cette commande, le programmeur fait l'économie d'un PRINT.

Exemple : !CHR$ 12,10,151,132,67,69,79,45,77,97,103

Efface l'écran, saute une ligne puis écrit "CEO-Mag" en bleu sur fond blanc.

!CLI

Inhibe les interruptions et empêche la lecture clavier.

Cette commande équivaut à la commande KEY OFF sous SEDORIC.

Notes : un !CLI augmente de 20% la rapidité de l'Oric. La seule manière de récupérer la main après un !CLI en mode direct est d'appuyer sur le bouton RESET.

!DATA AD_1, AD_2, N_1, N_2

Extrait la zone mémoire ad_1-ad_2 et recopie chaque octet (traduit en hexadécimal) dans le programme Basic à partir de la ligne numéro n_1. Chaque ligne de n_2 données créée contient une somme de contrôle.

Par défaut, la valeur n_2 est fixée à 10 si omise.

Cette commande est particulièrement utile pour la création de chargeurs de données en Basic et le partage de programmes en langage machine par revue interposée (CEO-Mag pour ne pas le nommer !).

Exemple :

10 !DATA #400,#4FF,1000,8

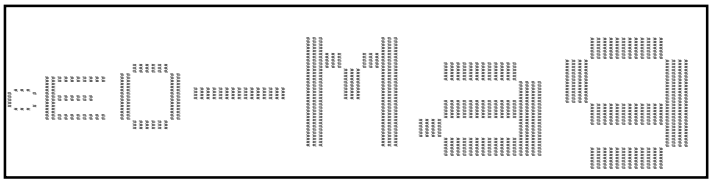
Créé des lignes de 8 données à partir du numéro de ligne 1000, pour la zone mémoire #400-#4FF (voir résultat pour les trois premières lignes ci-dessous).

```
1000 DATAC9,30,90,4,C9,3A,90,35,#355
1010 DATA86,F,AA,30,2E,85,C1,68,#34B
1020 DATAAA,68,48,E0,E,D0,4,C9,#3E5
```

!LPRINT CHAINE

Imprime la variable alphanumérique chaîne dans la taille x (de 1 à 10).

```
10 TEXT:CLS
20 !INPUT@2,3;"Entrez votre texte ";N$
30 N$=LEFT$(N$,10)
40 !PRINT@3,6;"Appuyez sur une touche quand"
50 !PRINT@4,8;"l'imprimante est prete"
60 GETZ$
70 !PRINT ON
80 FORI=1TOLEN(N$)
90 !LPRINT MID$(N$,I,1),I
100 NEXT
110 !PRINT OFF
```



!PRINT ON/OFF

Agit sur la mise en service ou l'arrêt de l'imprimante.

Le traitement du positionnement par '@' est conservé de manière classique.

!READY CHAINE

Affiche le message qui est dans chaîne en lieu et place du classique 'Ready'.

La chaîne ne doit pas excéder 27 caractères (?STRING TOO LONG ERROR dans le cas contraire) et ne doit pas être nulle (sinon ?SYNTAX ERROR).

!REM CO,L_1,L_2

Place le code couleur co après chaque REM trouvé entre les lignes l_1 à l_2 (mode direct uniquement).

$$0 \leq \text{code couleur} < 9.$$

En cas d'omission des lignes l_1 et l_2, l'intégralité du code Basic est passé en revue par la commande !REMco.

Le fait de spécifier la ligne de départ l_1 oblige de spécifier une ligne de fin l_2 (qui peut ne pas exister).

!SEI

Rétablit les interruptions et du même coup la lecture clavier.

Cette commande équivaut à la commande KEY SET sous SEDORIC.

!STOP R

Stoppe le programme et saute l'instruction suivante sans afficher le message BREAK IN n° ligne.

La reprise du programme se fait de manière habituelle grâce à la commande `CONT`.

Cette commande efface (Reset) le précédent message de prompt définit par la commande `!READY` "message".

!VERIFY CLOAD/CSAVE

Commande ORIC 1 uniquement.

Permet de vérifier si un programme a été sauvegardé correctement sur la cassette.

Fournit également des informations sur le type de données (Basic / LM), le mode de sauvegarde (Auto / Stop) et les adresses de début et de fin de programme (redondant avec la commande !CAT).

CLASSEMENT ALPHABETIQUE DES COMMANDES BASIX

Ci-dessous la syntaxe exacte de chaque commande classée par ordre alphabétique, ainsi que la référence du CEO-Mag dans lequel cette commande a été passée à la loupe.

INSTRUCTION	EXPLICATION	N° CEO-Mag
AND x_1,x_2,y_1,y_2,n	Remplit un rectangle qui débute en x_1,y_1 et qui s'achève en x_2,y_2 avec la valeur 'AND n'.	257
BOX n_1,n_2,fb	Trace un rectangle de n_1 pixels sur n_2 pixels avec le code fb. Le point de départ du tracé est le coin supérieur gauche du rectangle.	260
CAT "nom programme"	Affiche les caractéristiques liées à l'en-tête de "nom programme" (début, fin, AUTO, Basic / LM, nom programme).	264
CHAR A$,X,Y,$n_1$,$n_2$,H ou V	Ecrit une chaine de caractères en haute résolution.	259
CHR$ n_1,n_2,n_3,…n_i	Envoi à l'écran les caractères n_1 à n_i. Avec cette commande, le programmeur fait l'économie d'un PRINT.	267
CLI	Inhibe les interruptions et empêche la lecture clavier.	266
COL x,n	Place la valeur n dans la colonne x (compris entre 0 et 39). Fonctionne en mode TEXT ou en mode HIRES.	257
DATA ad_1,ad_2,n_1,n_2	Extrait la zone mémoire ad_1-ad_2 et recopie chaque octet (traduit en hexadécimal) dans le programme Basic à partir de la ligne numéro n_1. Chaque ligne de n_2 données créée contient une somme de contrôle. Par défaut, la valeur n_2 est fixée à 10 si omise.	267
DEEK adr	Affiche les valeurs stockées en mémoire, à partir des deux emplacements adr et adr+1 et ce jusqu'à appui sur la combinaison de touches CTRL + 'C'. L'appui sur 'espace' permet de stopper momentanément l'affichage des valeurs à l'écran.	263
DEF "étiquette"	Permet de placer une étiquette en début de ligne Basic.	262
DEFCAR x_1,n_1,n_2,n_3,n_4,n_5,n_6,n_7,n_8;x_2,..n_i	Sauve les valeurs n_1 à n_8 à l'emplacement #B3FF+x_i*8 (mode TEXT). X_i compris entre 0 et 9.	264
DEG n	Définit la valeur en degrés qui sera utilisée pour la commande FORWARD.	258
DELETE n_1,n_2	Détruit les lignes du programme Basic comprises entre n_1 et n_2 (incluses).	262
DELREM n_1,n_2	Détruit les lignes du programme Basic comportant des REM, de la ligne n_1 à la n_2.	262
EDIT n	Edite la ligne Basic n (ce qui en réalité revient au même que la commande Basic EDIT…).	262
FN	Définition de fonctions récursives.	271

INSTRUCTION	EXPLICATION	N° CEO-Mag
FORWARD n,fb	Réalise un tracé de n pixels (depuis la valeur courante) selon le code fb et l'angle définit par DEG. Equivaut à la commande Sedoric LINE n,fb.	258
GOSUB "étiquette"	Se rend à la ligne comportant l'étiquette définit précédemment par la commande DEF.	262
GOTO "étiquette"	Idem GOSUB "étiquette".	262
HDEEK adr	Fonctionne de la même manière que DEEK adr avec des valeurs hexadécimales.	263
HELP	Affiche la liste des commandes BasiX actuellement exploitables. Le listing peut être mis en pause à tout moment par appui sur la touche `espace´ puis stoppé par la combinaison CTRL + C.	262
HIRES	Provoque l'impression de l'écran HIRES. L'impression n'est possible que sur une véritable imprimante, le code émis n'étant pas exploitable dans un fichier "printer.txt".	271
HPEEK adr	Fonctionne de la même manière que PEEK adr avec des valeurs hexadécimales. Cette commande s'apparente fortement à un DUMP mémoire.	263
INPUT @x,y;"message";var	Positionne la commande INPUT aux coordonnées x,y.	267
INV x_1,x_2,y_1,y_2	Inverse vidéo de la surface délimitée par x_1,y_1 à x_2,y_2.	257
LEN n	Définit la longueur maximale d'une ligne Basic (n ne peut être nul).	262
LINE fb,(x_1,y_1 TO x_2,y_2)	Trace une ligne des coordonnées x_1,y_1 aux coordonnées x_2,y_2 avec le code fb.	260
LIST expression	Recherche et liste l'expression demandée. Cette commande s'apparente fortement à la commande SEEK sous SEDORIC à la différence près que le caractère joker '£' n'est pas pris en charge.	264
LOMEM adr	Fixe le début des programmes Basic à l'adresse adr. Une adresse supérieure à celle fixée par la commande Basic HIMEM provoque un ?ILLEGAL QUANTITY ERROR.	263
LPRINT chaîne,x	Imprime la variable alphanumérique chaîne dans la taille x (de 1 à 10).	266
MAX MA=a,b,c,d,…	Détermine la valeur maximum d'une suite de nombres et l'attribue à la variable MA.	261
MERGE "nom programme"	Fusionne le programme demandé (au format TAP) avec celui en mémoire. Lors de la fusion, le message "Working .." s'affiche sur la ligne de statut.	262
MIN MI=a,b,c,d,…	Détermine la valeur minimum d'une suite de nombres et l'attribue à la variable MI.	261

INSTRUCTION	EXPLICATION	N° CEO-Mag
MODIFY $expression_1$,$expression_2$(,$ligne_1$,$ligne_2$)	Réalise le remplacement de l'$expression_1$ par l'$expression_2$ dans le programme Basic, de la $ligne_1$ à la $ligne_2$ (facultatif).	264
MOVE adr_1,adr_2,n	Copie la zone mémoire (adr_1+n) à partir de l'adresse adr_2.	263
OLD	Récupère un programme Basic effacé par NEW.	262
ON i GOTO exp_1,exp_2,…,exp_i ON i GOSUB exp_1,exp_2,…,exp_i	Se rend à ligne Basic exp_i (GOTO). Se rend au sous-programme situé en ligne exp_i (GOSUB). L'expression exp_i peut être numérique ou alphanumérique. Dans ce dernier cas, elle devra être déclarée par la commande DEF.	270
OR x_1,x_2,y_1,y_2,n	Remplit un rectangle qui débute en x_1,y_1 et qui s'achève en x_2,y_2 avec la valeur 'OR n'	257
PACK AD,AF	Compacte l'écran HIRES à l'adresse AD	259
PAINT x,y	Remplit une surface à partir des coordonnées x et y.	257
PEEK adr	Fonctionne de la même manière que DEEK adr, mais à partir du seul emplacement adr.	263
POKE adr_1,adr_2,n_1,n_2,n_3,n_i,…	Place les valeurs n_1 à n_i à partir de l'adresse adr_1, puis répète la séquence n_1 à n_i jusqu'à l'adresse adr_2.	263
POP	Définition de fonctions récursives.	271
PRINT ON/OFF	Agit sur la mise en service ou l'arrêt de l'imprimante. Le traitement du positionnement par '@' est conservé de manière classique.	264
RANDOM	Initialise le générateur de nombres aléatoires au hasard.	262
READY chaîne	Affiche le message qui est dans chaîne en lieu et place du classique 'Ready'.	266
REM co,l_1,l_2	Place le code couleur co après chaque REM trouvé entre les lignes l_1 à l_2 (mode direct uniquement). $0 \leq$ code couleur < 9.	267
RESTORE n	Place le pointeur des DATA à la ligne n.	262
RETURN	Définition de fonctions récursives.	271
ROT n	Réalise une rotation de n degrés.	258
ROUND var,expression	Arrondit la variable 'var' d'un nombre équivalent au résultat de 'expression'. 'var' est de type réel.	264
SCREEN n_1,n_2,n_3,n_4	Créé un sous-écran texte de n_2 lignes à partir de la ligne n_1, avec les couleurs PAPER n_3 et INK n_4.	260
SEI	Rétablit les interruptions et du même coup la lecture clavier.	266
SET val TO var_1,var_2,var_3,…var_i	Initialise les variables var_1 à var_i avec la valeur val (variables numériques exclusivement).	267
SORT tableau(0)	Tri le tableau de la plus petite à la plus grande valeur.	261

INSTRUCTION	EXPLICATION	N° CEO-Mag
STOP R	Stoppe le programme et saute l'instruction suivante sans afficher le message BREAK IN n° ligne. La reprise du programme se fait de manière habituelle grâce à la commande CONT.	271
SWAP var_1,var_2	Intervertit les contenus respectifs des deux variables var_1 et var_2 de même type (dans le cas contraire, vous avez droit à un ?TYPE MISMATCH ERROR).	263
TROFF	Désactive le gestionnaire d'erreurs Basic.	262
TRON	Active le gestionnaire d'erreurs Basic.	262
UNPACK AD	Décompacte l'écran HIRES n° AD	259
VAL var = A$	Transfère dans la variable 'var' le résultat de l'expression numérique contenu dans A$ (longueur maximum de 79 caractères).	265
VERIFY CSAVE VERIFY CLOAD	**Commande ORIC 1 uniquement.** Permet de vérifier si un programme a été sauvegardé correctement sur la cassette. Fournit également des informations sur le type de données (Basic / LM), le mode de sauvegarde (Auto / Stop) et les adresses de début et de fin de programme (redondant avec la commande !CAT).	N/A
VLIST	Liste les variables en cours d'utilisation au moment du traitement d'une erreur.	262
VPUT expr TO adr	Stocke le résultat de l'expression numérique 'expr' à l'adresse 'adr'.	265
WHERE AD,AF	Fonctionne de manière identique à PACK mais sans stockage. Cette commande présente l'avantage de déterminer quelle serait l'adresse de fin de l'écran compacté et de s'assurer ainsi que l'on ne risque pas d'écraser la zone de stockage de BasiX .	259