L'ORIC A NU

FABRICE BROCHE

SORACOM éditions
INFORMATIQUE

L'ORIC A NU

FABRICE BROCHE

L'ORIC A NU

SORACOM éditions
INFORMATIQUE

INTRODUCTION

A) INTRODUCTION

1-But de l'ouvrage

Ce livre est unique en son genre: Il se propose d'aller au plus profond de la structure logicielle d'un ordinateur,en commentant toute sa mémoire morte,la ROM BASIC.

Ce commentaire se situe à deux niveaux:

-Commentaire du code machine,ligne par ligne, et ce pour les 16 K. Chaque routine est expliquée,pour rendre sa compréhension et son utilisation accessible à tous. Les routines les plus compliquées sont accompagnées d'organigrammes.

-Chapitres généraux, faisant le point sur un thème précis: la gestion des entrée/sortie, le fonctionnement d'un interpréteur etc.

Et c'est là la nouveauté par rapport aux ouvrages du genre, quel que soit l'ordinateur considéré. Il existait jusqu'à présent deux types d'ouvrages: ceux qui donnaient un listing commenté de la ROM, sans autre explication, ce qui n'est utile que pour celui qui connait déjà le principe des routines employées, ou ceux qui sont une sorte de recueuil de connaissances sur une machine, ils ne sont alors pas exhaustifs, et décrivent l'utilisaion plus que le fonctionnement.

Cet ouvrage fait donc la synthèse de tout ce que l'on peut savoir sur l'Oric.

Sa lecture ne requiert que peu de connaissances particulières: il suffit de connaitre le jeu d'instruction du 6502, et d'avoir le gout de comprendre ce qui se passe.

A ce sujet, il n'est de meilleur apprentissage du langage assembleur que d'analyser des programmes écrits très correctement: l'effet des mnémoniques s'assimile tout seul au fur et à mesure de la lecture de la ROM.

Une dernière précision: ce livre s'adresse surtout aux possesseurs d'Oric-1 et Atmos, mais il intéressera tous les utilisateurs de Basic Microsoft: les routines sont exactement les mêmes, à l'emplacement près. les possesseurs de COMMODORE 64,APPLE II,VIC 20 etc.. sont donc concernés.

2-Comment lire le livre

La lecture peut se faire à deux nivaux

-Niveau consultation: on veut juste savoir comment utiliser telle ou telle routine,ou savoir s'il il existe une routine qui... Il sufit de se reporter au répertoire des routines, classées par thèmes,et de consulter l'en-tête des routines, qui indique les paramètres en entrée, et les résultats retournés.

-Niveau approfondissement des connaissances: il s'agit cette fois d'une lecture attentive des routines, du principe de fonctionnement etc..

Il est dans tous les cas conseillé de parcourir l'ensemble du livre, pour savoir où chercher le renseignement souhaité.

L'ensemble des commentaires est conçu de telle sorte que le lecteur qui a lu attentivement tout l'ouvrage doit être capable d'écrire un BASIC.

A part les quelques notices techniques dont la liste est donnée en annexe, tous les renseignements et commentaires sont le fruit de recherches personnelles. Tout ce qui est avancé dans cet ouvrage a été scrupuleusement vérifié, mais des erreurs peuvent bien entendu subsister.

B) CONVENTIONS ET NOTATIONS

Pour éviter d'alourdir le commentaire, un certain nombre de conventions ont été adoptées.

1-Structure du commentaire

Le début d'un routine est repéré par un titre centré et en majuscules.

Des séparations logiques à l'intérieur d'une routine sont repérées par un titre non centré et en minuscules.

Le titre décrit le plus précisément possible l'action globale de la routine.

Un titre comprend parfois des mots entre parenthèses:

Par exemple,COMMANDE: il s'agit de la routine d'exécution d'une commande du BASIC.

Ex:'RUN' (COMMANDE)

De même, FONCTION et OPERATEUR indiquent l'exécution d'une fonction ou d'un opérateur.

Les commentaires généraux sur la routine sont repéres à l'aide de noms génériques:

Entrée: désigne les paramètres particuliers qui doivent être positionnés correctement pour avoir les résultats escomptés.

Sortie: désigne les paramètres retournés par la routine,ainsi que quelques renseignements indirectement liés, mais utilisés dans la ROM, ou utilisables, du type registre inchangé, indicateur positionné etc...

Programmation: des remarques sur les astuces employées, ou sur les manques d'optimisation. Ce genre de remarque est surtout fait dans le corps du commentaire.

Principe: explique le principe général de la routine, éventuellement aidé par des organigrammes ou divers schémas.

Bogue: signale les erreurs de programmation qui font que la routine ne donne jamais (ou pas toujours) le résultat escompté. Elles sont plus nombreuses qu'on le croit !

D'autres noms génériques sont parfois employés, ils parlent alors d'eux même et il est inutile de s'étendre sur leur signification.

La structure du listing est peu classique, mais très pratique: il donne les deux ROM's en parallèle, sachant que bien souvent, les routines sont les mêmes, mais décalées simplement. Là où les ROM's différent, des points de suspension remplacent le listing (en aucun cas la linéarité du listing n'est rompue).

Le premier listing est toujours celui de la V1.Ø,et le deuxième celui de la V1.1 .

Les mnémoniques sont bien entendu standards. Toutefois,les modes d'adressages ont une syntaxe particulière, adoptée pour réduire la place du listing et donc priviliéger le commentaire.

Voici les conventions utilisées pour les modes d'adressages:

```
LSR A          Adressage accumulateur
LDA #12        Immédiat: charge A avec le code hexa #12
LDA #'2'       Immédiat: charge A avec le code ASCII de 2,soit #32
LDA #&END      Immédiat: charge A avec le token du mot clé,soit ici #8Ø
STX Ø23Ø       Absolu (argument en hexadécimal)
ADC 91         Page Ø (argument en hexadécimal)
TXA            Implicite
LDA (12,X)     Pré indexé indirect (jamais utilisé dans la ROM)
LDA (E9),Y     Post indexé indirect (argument en Hexadécimal)
STA 4Ø,X       Page Ø indexé
LDY Ø235,X     Absolu indexé
BNE C57Ø       Relatif
JMP $CØØØ      Saut
JMP (ØØ91)     Indirect
```

Un mnémonique spécial est utilisé: BYT, qui indique l'entrée d'octets,

d'adresse (stokées poids faible d'abord), ou de chaines de caractères.
Ex:C3AA C3A6 BYT ' ERROR'

Un commentaire revient souvent en regard des branchements relatifs: Inconditionnel. Il signifie que compte tenu de la position des indicateurs concernés, le branchement sera toujours effectué.

Une dernière précision: toutes les tables, ainsi évidemment que le listing, ont été générés par programme et non recopiées, ce qui devrait assurer un listing exempt d'erreur.

2-Abréviations

Tous les registres et indicateurs du 6502 sont repérés directement par leur lettre en majuscule:

A=Accumulateur
X=Index
Y=Index
P=Registre d'état
S=Pointeur de pile
Z=Indicateur de zéro
C=Retenue
N=Indicateur de signe
V=Indicateur d'overflow
I=Drapeau d'autorisation d'IRQ
B=Indicateur de BRK
D=Drapeau mode décimal

Un octet est représenté sous la forme: b7 b6 b5 b4 b3 b2 b1 b0. Ainsi, b3 signifie que l'on fait référence au bit 3 de l'octet.
Les accumulateurs flottants sont repérés par ACC1,ACC2,ACC3,ACC4,ACC5:

ACC1:#D0-#D5 (accumulateur principal)
ACC2:#D8-#DD
ACC3:#95-#98
ACC4:#C6-#CA
ACC5:#CB-#CF

De plus, les accumulateurs, après arrondi, sont notés AACC1, AACC2 etc...

Les valeurs contenues par les registres sont notées le registre poids faible d'abord.

Ex:YA --) ACC1 veut dire placer le nombre dont le poids faible est dans Y et le poids fort dans A dans l'accumulateur principal.

Lorsque des pointeurs mémoires sont utilisés, les poids forts et faibles sont séparés par un tiret.

Ex:#9A-#9B,#200-#201

Comme d'habitude, les parenthèses désignent une indirection: le contenu de l'adresse pointée par.

Ex: si A vaut #04 et Y vaut #C0, AY signifie #C004, YA signifie #04C0 et (AY) signifie le contenu de la case mémoire #C004.

Ce procédé a été étendu un peu abusivement pour les calculs en flottant: (AY) signifie les 5 octets débutant à l'adresse AY.

Ex: (AY) --) ACC1.Si A vaut 12 et Y vaut #45, cela signifie: transférer le nombre codé en virgule flottante dans la zone mémoire #4512-#4516 dans l'accumulateur flottant principal.

Ne disposant pas des labels standards, il n'était pas concevable de se lancer dans l'invention, et de donner des noms aux routines. Lorsque dans le texte il est fait référence à des routines ou pointeurs, ils sont repérés par leurs adresses. Dans le cas d'une routine d'adresse différente pour les deux Roms, les 2 adresses sont données,séparées par un '/'

Ex:#F276/#F30A.

Une seule exception:le pointeur du texte BASIC est noté TXTPTR dans les commentaires (mais pas dans le listing). TXTPTR=#00E9.

CHAPITRE II

ORGANISATION MATERIELLE

A) PRESENTATION GENERALE

1-Généralités

L'ORIC est architecturé autour d'une untité centrale (CPU pour Central Procesing Unit) 6502 A tournant à 1 MHz.

La structure est classique: CPU, ROM, RAM, circuits d'E/S.

Un micro ordinateur est un curieux mélange de composants très puissants qui comportent des dizaines de milliers de portes logiques (CPU 6502, RAM etc...), et de composants très peu puissants: 4 ou 8 portes seulement, sans compter les éléments discrets.

En effet, il faut relier tout ces composants principaux entre eux.

On peut alors distinguer (assez artificiellement) les Bus (Bus de données qui véhiculent les informations, et Bus d'adresse qui indiquent la destination ou la source de l'information), et les lignes de controle.

Les bus sont simples à mettre en oeuvre. Toutefois, il est souvent nécessaire de les amplifier pour éviter une trop grande déperdition dans les boitiers. Les bus de l'Oric ne sont pas amplifiés, ce qui est génant lorsque l'on branche des extensions.

Les lignes de controle regroupent tout ce qui a trait à la validation des données présentes sur le Bus (c'est le cas de l'horloge) ou à la validation des différents boitiers susceptibles d'être adressés.

Ainsi, la plupart des composants ont une ou plusieurs lignes 'CS', abréviation de Chip Select, validation du boitier. Etant donné la manière dont le 6502 gère son espace adressable, chaque boitier doit être sélectionné pour une zone mémoire particulière, et il faut bien sur éviter que deux boitiers soient sélectionnés en même temps.

Le plus gros travail consiste à effectuer un décodage sur le bus d'adresse: sélectionner la ROM uniquement entre #C000 et #FFFF, ou le VIA entre #300 et #3FF. Un tel décodage est très facile à faire, mais nécessite un nombre important de portes logiques.

Il faut aussi s'occuper de générer une image vidéo, c'est à dire lire le contenu de l'écran en mémoire et transformer ces informations en un signal couleur. Il faut enfin 'rafraichir' la RAM, qui est de type dynamique (on

peu schématiser une cellule RAM par une capacité chargée pour un 1 et déchargée pour un Ø), pour éviter de perdre son contenu.

Toutes ces opérations sont relativement simples mais nécessitent un très grand nombre de composants.

C'est pourquoi les concepteurs ont choisi la technologie de l'ULA (Uncommited Logic Array, réseau logique programmable), qui est un circuit comportant un grand nombre de portes logiques, que le concepteur organise selon ses besoins. Il y a quelques années, une centaine de circuits auraient été nécessaires pour réaliser le travail de l'ULA (ne pas confondre avec UAL, unité arithmétique et logique, ALU en anglais).

Aucune documentation n'a filtré, hélas, sur l'ULA. Oon est donc réduit à son brochage et beaucoup de suppositions. On sait toutefois globalement son action:

-Lire la mémoire vidéo et générer les signaux RVB et synchro.

-Décoder le bus d'adresse pour générer les signaux de validation de la ROM, du VIA et de la RAM.

-Faire une lecture 'bidon' de la RAM pour la rafraichir. Etant donné l'organisation de la RAM, il est possible que le rafraichissement vidéo suffise à rafraichir la RAM. Sous toutes réserves.

Le 65Ø2 n'utilisant pas le Bus tout le temps, l'ULA utilise les temps morts pour travailler, ce qui donne des timings très serrés (Cf MAP par exemple). Elle dispose pour cela d'un horloge à 12 Mhz, à partir de laquelle elle génère un signal d'horloge à 1 Mhz pour le 65Ø2 , le VIA etc...

2-Synoptique

Voici le schéma de principe de l'Oric:

(voir page 13)

Peu de choses à signaler, sinon une utilisation optimale du VIA 6522 pour gérer les entrées sorties.

3-Schéma de l'oric

Voici maintenant le schéma de l'Oric 48K dans sa dernière version (ISSUE 4).

(voir page 14)

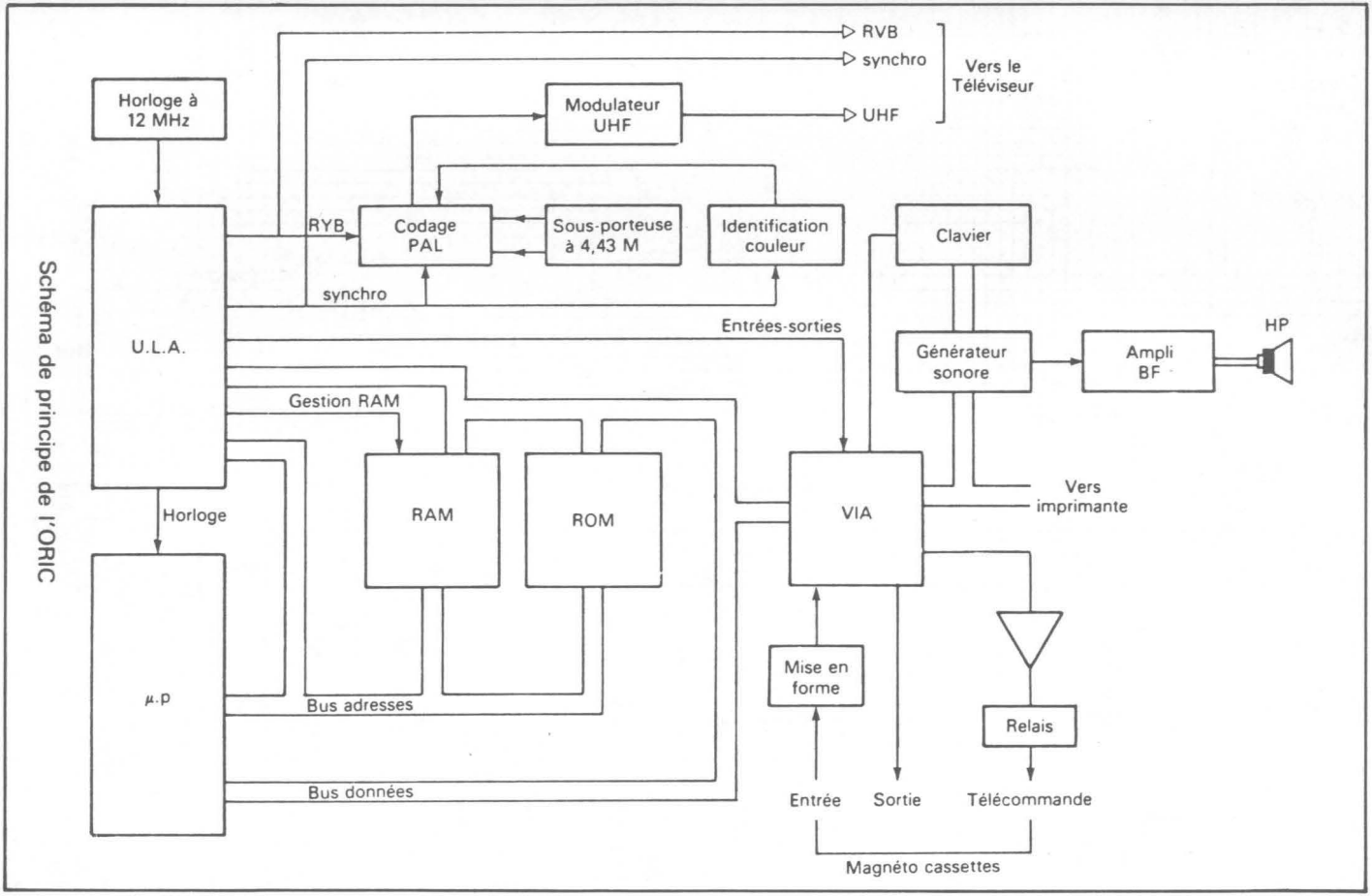
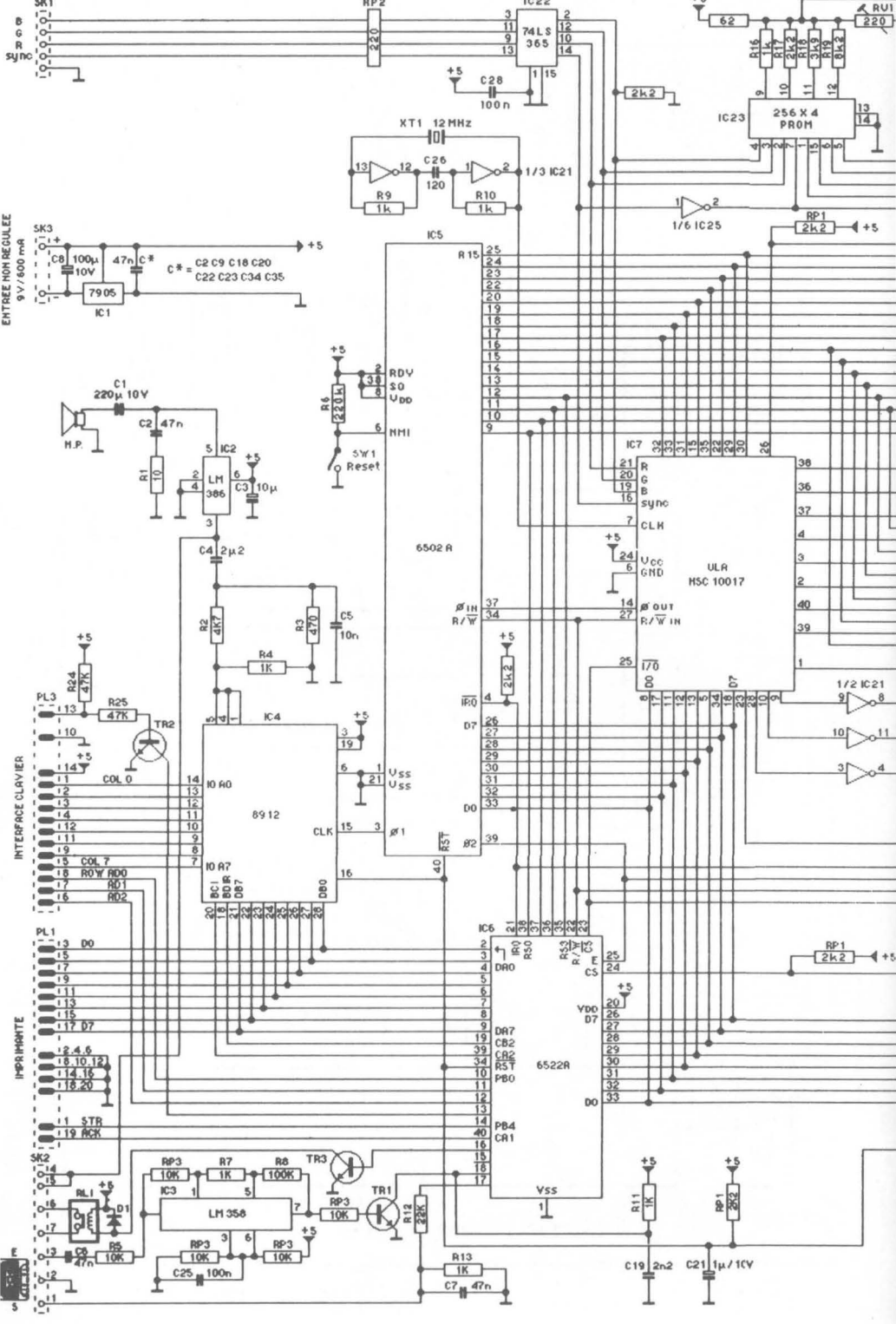


Schéma de principe de l'ORIC
RVB
synchro
UHF
Vers le Téléviseur
Horloge à 12 MHz
Modulateur UHF
RYB
Codage PAL
Sous-porteuse à 4,43 M
Identification couleur
Clavier
synchro
U.L.A.
Entrées-sorties
Générateur sonore
Ampli BF
HP
Gestion RAM
RAM
ROM
VIA
Vers imprimante
Horloge
μ.p
Bus adresses
Mise en forme
Relais
Bus données
Entrée
Sortie
Télécommande
Magnéto cassettes


SK1
RP2
220
IC22
74LS 365
C28
100n
+5
2k2
IC23
256 X 4 PROM
XT1 12 MHz
C26
120
R9
1k
R10
1k
1/3 IC21
1/6 IC25
RP1
2k2
SK3
ENTREE NON REGULEE 9V/600 mA
C8 100µ 10V
47n
C* = C2 C9 C18 C20 C22 C23 C34 C35
IC1
7905
IC5
6502 A
RDY
SO
VDD
NMI
SW1
Reset
C1
220µ 10V
M.P.
C2 47n
R1 10
IC2
LM 386
C3 10µ
C4 2µ2
R2 4K7
R3 470
C5 10n
R4 1K
IC7
ULA
MSC 10017
VCC
GND
CLK
sync
R24 47K
R25 47K
TR2
PL3
COL 0
IC4
8912
INTERFACE CLAVIER
COL 7
ROW AD0
AD1
AD2
PL1
IMPRIMANTE
STR
ACK
IC6
6522A
SK2
RL1
D1
C9 47n
R5 10K
RP3 10K
R7 1K
R8 100K
IC3
LM 358
TR3
TR1
R12 22K
C25 100n
R13 1K
C7 47n
R11 1K
C19 2n2
C21 1µ / 10V
1/2 IC21

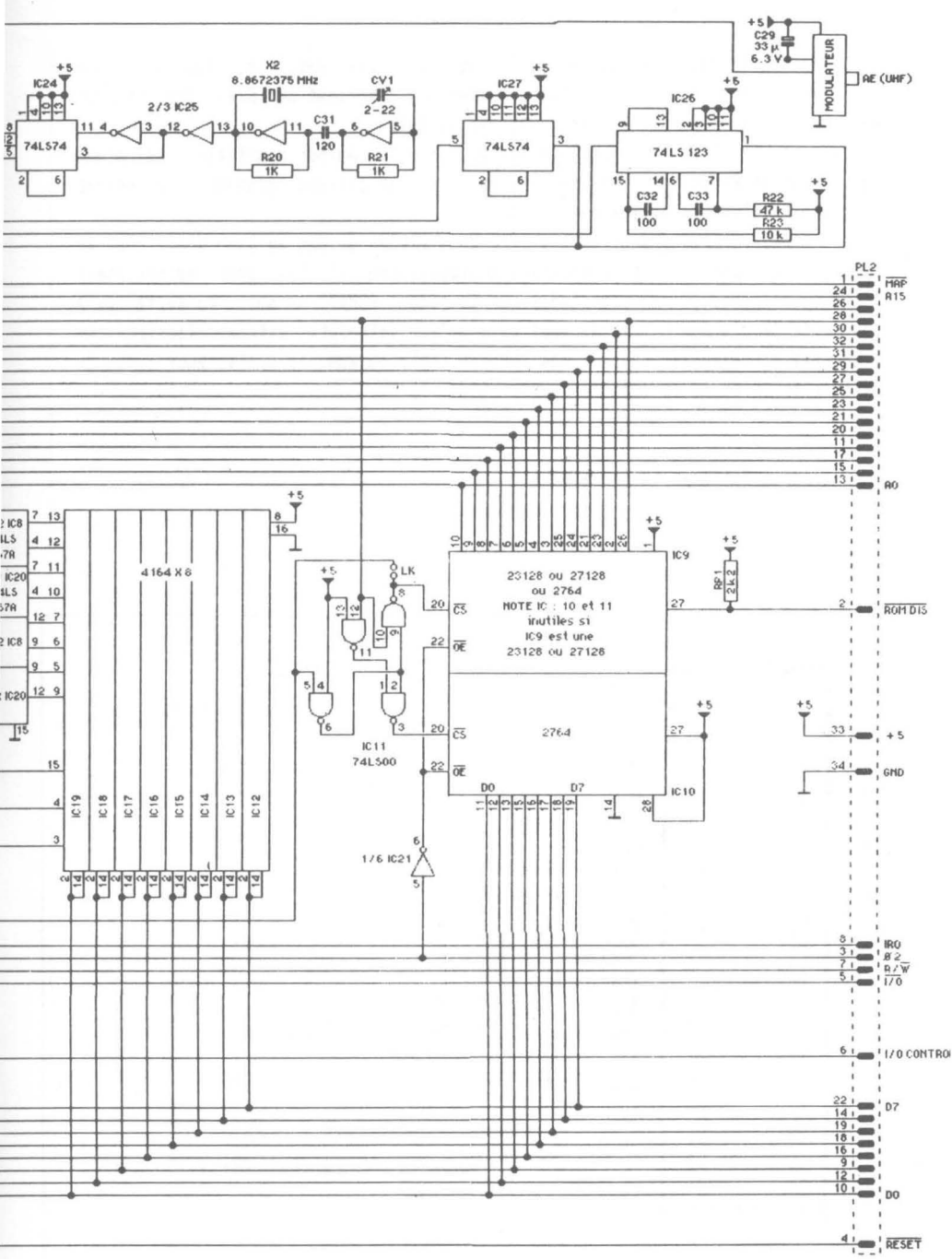


Schéma de l'ORIC 48 k

Signalons le fait regrétable que le bus n'est pas amplifié, pas plus que certaines lignes de controle vitales telles que l'horloge ou R/W. Ces signaux sont très faibles lorsqu'ils présents sur le Bus.

Le poussoir 'NMI' est traité de manière pour le moins spartiate, et n'est pas exempt de rebond, ce qui explique qu'il faille parfois plusieurs pressions pour sortir d'une situation désespérée.

Les premiers Oric-1 sont sortis avec deux EPROMS de 8Ko au lieu d'une ROM de 16 Ko, ce qui nécessitait un décodage suplémentaire (IC 11) pour sélectionner l'une ou l'autre. Il est regrétable que le signal ROMDIS ne soit connecté qu'à une seule de ces ROM's, ce qui veut dire qu'une interface externe (lecteur de disquette par exemple) ne peut pas inhiber toute la ROM. Il suffit pour celà de diriger le CS de IC 10 vers le ROMDIS et non le 5V.

4-caractéristiques du CPU 6502 A

Yss	1	40	RES
Rdy	2	39	O2
O1	3	38	SO
IRQ	4	37	O0
N.C	5	36	N.C
NMI	6	35	N.C
SYNC	7	34	R/W
Ycc	8	33	D0
A0	9	32	D1
A1	10	31	D2
A2	11	30	D3
A3	12	29	D4
A4	13	28	D5
A5	14	27	D6
A6	15	26	D7
A7	16	25	A15
A8	17	24	A14
A9	18	23	A13
A10	19	22	A12
A11	20	21	Yss

RES:RESET
NMI:NMI
IRQ:IRQ (!)
O0:entrée horloge
O1:sortie horloge O1=O0+pi
02:sortie horloge O2=O1
A0-A15: bus d'adresse
D0-D7: bus de donnée
SO: un front négatif positionne V
SYNC: à 0 pendant la lecture du code opération

BROCHAGE 6502-A

MODELE DE PROGRAMMATION DU 6502

A :									b7	b6	b5	b4	b3	b2	b1	b0
X :									b7	b6	b5	b4	b3	b2	b1	b0
Y :									b7	b6	b5	b4	b3	b2	b1	b0
P :									S	V	?	B	D	I	Z	C
SP:	0	0	0	0	0	0	0	1	b7	b6	b5	b4	b3	b2	b1	b0
PC:	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

S: signe. Reflet de b7 lors des opérations sur A
V: Overflow. Propagation de b6 vers b7
B: Break (ou intéruption simulée)
D: Mode décimal
I : Autorisation des IRQ
Z: Résultat nul
C: Retenue (pour Carry)

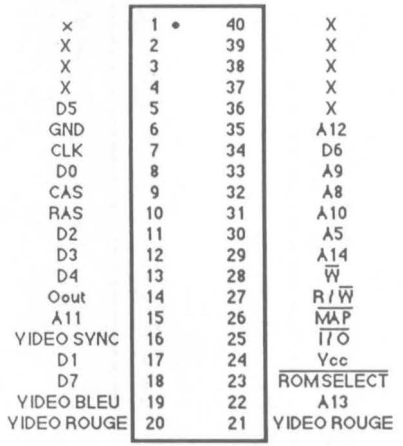
Tous ces indicateurs sont actifs si ils sont à 1

NB: Tous les registres A,X,Y et SP se comportent "modulo 256", c'est à dire qu'ils passent de #FF à 0 ou de 0 à #FF.

5-caractéristiques de l'ULA HCS 10017

Figure ULA-A

BROCHAGE ULA HCS 10017



D0-D7: Bus de données
A8-A15: Bus d'adresse
CLK: horloge entrée 12 MHz
CAS: RAM, selection colonne
RAS: RAM, sélection ligne
Oout: horloge sortie 1 MHz
VIDEO: RGB synchro
W: lecture : écriture RAM
R/W: lecture / écriture RAM
I/O: #300-#3FF
ROM SELECT: #C000-#FFFF
MAP: Validation RAM

B) LES ENTREES / SORTIES.

1-Généralités

L'oric utilise essentiellement deux composants pour dialoguer avec le monde extérieur (outre l'ULA pour la vidéo): le VIA 6522 (Versatile Interface Adaptator, soit circuit d'interfa'cage multi usages) et le PSG 8912 (Programmable Sound Generator, soit générateur de sons programmables).

Leur utilisation est quasi optimale, puisque une ligne seulement du VIA n'est pas utilisée !

2-Schéma des entrées/sorties (sauf vidéo)

Voici un schéma résumant la configuration de ces boitiers.

(voir page 19)

Schéma A

Voici un résumé des caractéristiques des boitiers entrant en jeu, accompagné d'un modèle de programmation, indispensable pour comprendre complètement les routines d'E/S de la ROM.

3-Caractéristiques du VIA 6522 A

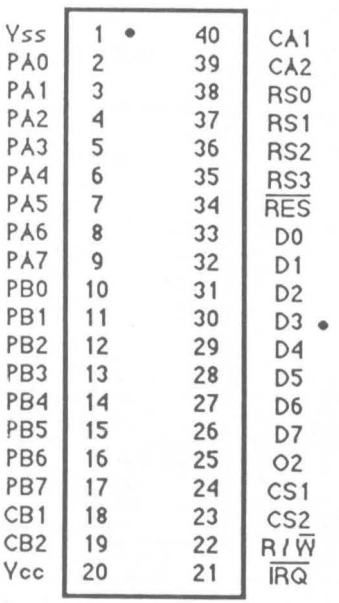
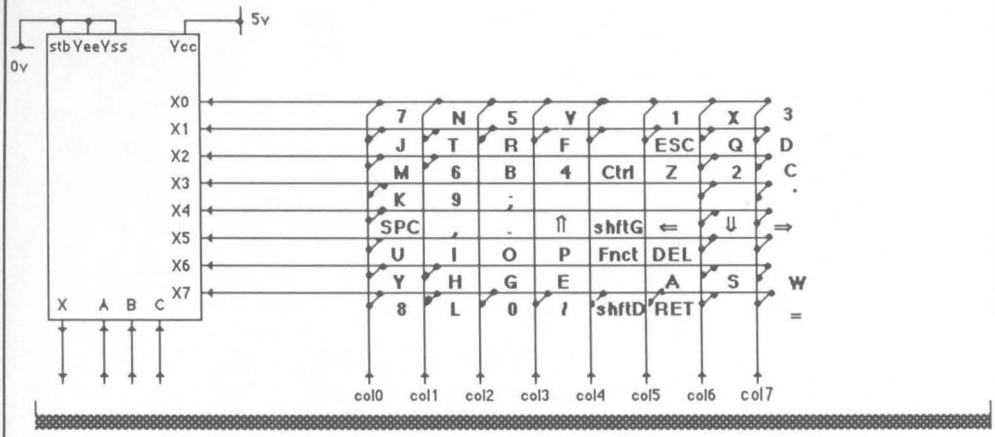


Figure VIA-D

PA0-PA7: port d'entrée / sortie A
PB0-PB7: port d'entrée / sortie B
RS0-RS3: sélection des registres
CS1-CS2: chip select
RES : RESET



CONNECTEUR CLAVIER COTE CLAVIER

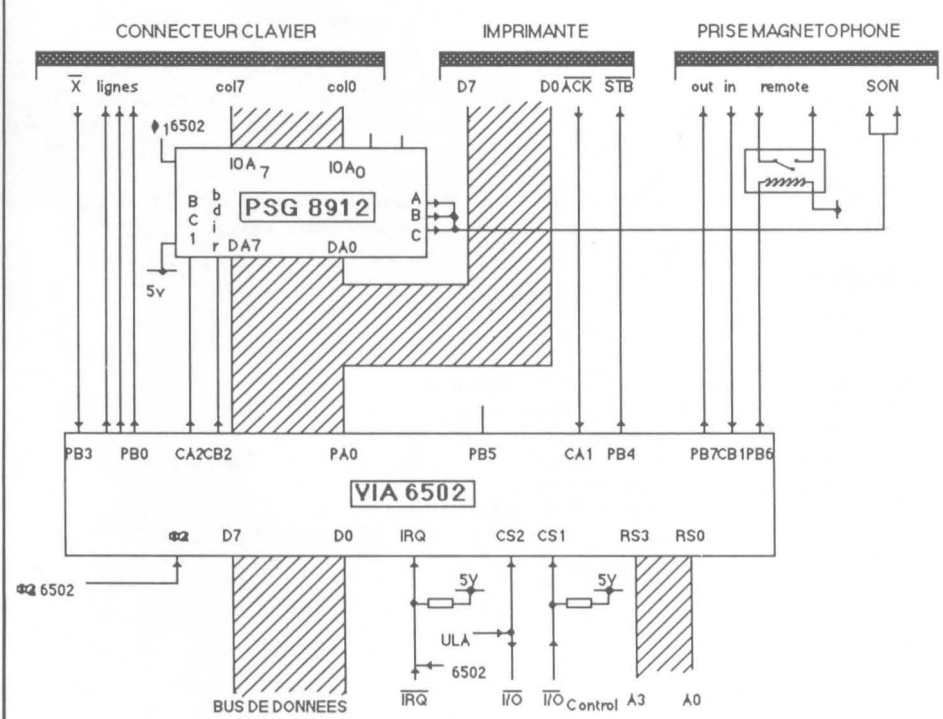


Schéma A

Le VIA est vu par le 6502 comme 16 registres consécutifs:

00: VIADRB	08: VIAT2L
01: VIADRA	09: VIAT2H
02: VIADDRB	0A: VIASR
03: VIADDRA	0B: VIAACR
04: VIAT1L	0C: VIAPCR
05: VIAT1H	0D: VIAIFR
06: VIAT1LL	0E: VIAIER
07: VIAT1LH	0F: VIAORA / VIAIRA

VIADDRA (3) et **VIADDRB (2)**

Commande de la direction des données. 1, la ligne est en sortie. 0, la ligne est en entrée.
Exemple, si on met #FE dans VIADDRA, PA0 est une entrée et PA1, PA2, PA3, PA4, PA5, PA6, PA7 sont des sorties.

VIADRA (1) et **VIADRB (0)**

Ces registres permettent de lire et écrire les données. les données sont latchées en sorties, c'est à dire qu'elles gardent leurs valeurs jusqu'au prochain changement.
L'entrée est éventuellement latchée elle aussi (voir VIAACR).
La lecture ou écriture de ces registres agissent sur les lignes de handshake (CA1, CB1, CA2, CB2) et sur les indicateurs d'IRQ correspondants conformément à VIAPCR.

VIASR (A)

C'est le registre de données du registre à décalage. Inutilisable sur l'ORIC car les données sont lues ou écrites via CB2, qui est utilisé pour autre chose.

VIAACR (B)

7 bits: MPB7 MT1 MT2 SR2 SR1 SR0 LB LA

LA autorisation du latche en entrée sur le port A
LB autorisation du latche en entrée sur le port B
SR2, SR1, SR0 commande de VIASR, inutile sur l'Oric.
MT2 0, décrémentation de T2 selon O2. 1, décrémentation de T2 selon PB6
MT1 0, mode monostable. 1, mode roue libre
MPB7 1, sortie autorisée sur PB7. 0, sortie interdite.
(si la sortie est autorisée, le niveau de PB7 est inversé quand T1=0)

Figure VIA-A

VIAPCR (C)
Le registre de controle périphérique contient 8 bits: B2 B1 B0 FB A2 A1 A0 FA

FA 1, détection de front montant sur la broche CA1. 0, détection de front négatifs.

A2	A1	A0		
0	0	0	CA2 entrée	!
0	0	1	CA2 entrée indépendante	! front descendant
0	1	0	CA2 entreé	!
0	1	1	CA2 entrée indépendante	! front montant
1	0	0	CA2 sortie, à 0 sur lecture / écriture de VIADRA	
1	0	1	CA2 sortie, impulsion sur lecture / écriture de VIADRA	
1	1	0	CA2 sortie, à 0	
1	1	1	CA2 sortie, à 1	

FB,B2,B1,B0 idem à ci-dessus mais pour le port B.

VIAIFR (D)
Le registre d'indicateur d'intéruption comporte 8 bits:
IRQ T1 T2 CB1 CB2 SR CA1 CA2

	MIS A 1 par:	MIS A 0 par:
CA2	transition CA2	Lecture / écriture VIADRA
CA1	transition CA1	Lecture / écriture VIADRA
SR	8 décalages effectués	Lecture / écriture VIASR
CB2	transition CB2	Lecture / écriture VIADRB
CB1	transition CB1	Lecture / écriture VIADRB
T2	T2 = 0	Lecture VIAT2L ou écriture VIAT2H
T1	T1 = 0	Lecture VIAT1L ou écriture VIAT1H
IRQ	IRQ active et autorisée	traitement de l'IRQ

- On peut aussi mettre le bit de l'IRQ à 0 en écrivant un 1 dans le bit correspondant.
- La ligne d'IRQ est le reflet exact de b7. Elle restera donc à l'état bas tant que l'IRQ ne sera pas traitée.
- L'intérêt de VIAIRA / VIAORA est que la lecture ou écriture de ces registres n'affectent pas les flags d'intéruption de CA2 et CA1.

Figure VIA-B

Figure VIA-C

VIAIER (E)
Le registre d'autorisation d'intéruption comporte 8 bits utiles:

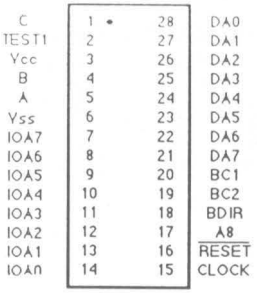
EN T1 T2 CB1 CB2 SR CA1 CA2

Ce registre se comporte assez bizarement, mais de manière en fait très pratique:
Pour autoriser une intéruption, il faut mettre simultanément à 1 le bit correspondant, mais aussi EN.
Pour interdireune intéruption, il faut mettre à 1 le bit correspondant, et mettre à 0 EN.
Tout ceci a pour conséquense que l'autorisation ou l'inhibition d'une intéruption n'affecte pas les autres.

VIAORA / VIAIRA (F)
Agit comme VIADRA mais ne modifie pas les status d'intéruption de CA1 et CA2.

Figure 8912-A

BROCHAGE 8912



A,B,C sorties analogiques
Test1: test du boitier par GI
IOA7-IOA0: port A
DA0-DA7: bus de données
RESET: reset, tous registres à 0
BC1,BC2,BDIR: sélection du boitier et de

BDIR	BC2	BC1	Mnémo	Fonction du PSG
0	0	0	NACT	Inactif, Cf 010
0	0	1	ADAR	Latcher adresse, Cf 111
0	1	0	IAB	Inactif. DA0-DA7 haute impédance
0	1	1	DTB	Lecture du registre courant
1	0	0	BAR	Latcher adresse, Cf 111
1	0	1	DW	Inactif, Cf 010
1	1	0	DNS	Ecriture du registre courant
1	1	1	INTAK	Latch adresse, Cf 001 et 100

Si on fixe BC2=1, la sélection se réduit à:

0	1	0	IAB	Inactif
0	1	1	DTB	Lecture registre
1	1	0	DNS	Ecriture registre
1	1	1	INTAK	Latcher adresse

Figure 8912-B

MODELE DE PROGRAMMATION 8912

R0 P7 P6 P5 P4 P3 P2 P1 P0 Période canal A
R1 X X X X P11 P10 P9 P8
P0-P15: Période (à multiplier par 16 pour avoir la période en micro-secondes)

R2 P7 P6 P5 P4 P3 P2 P1 P0 Période canal B
R3 X X X X P11 P10 P9 P8
P0-P15: Période (à multiplier par 16 pour avoir la période en micro-secondes).

R4 P7 P6 P5 P4 P3 P2 P1 P0 Période canal C
R5 X X X X P11 P10 P9 P8
P0-P15: Période (à multiplier par 16 pour avoir la période en micro-secondes)

R6 X X X X P3 P2 P1 P0 Période bruit mixable

R7 X IOA NC NB NA TC TB TA Autorisation
TA-TB-TC: autorisation des canaux musicaux (actif à 0)
NA-NB-NC: autorisation du mélange du bruit (actif à 0)
IOA: port A en entrée (0) ou sortie (1)

R8 X X X M L3 L2 L1 L0 Volume canal A
R9 X X X M L3 L2 L1 L0 Volume canal B
R10 X X X M L3 L2 L1 L0 Volume canal C
L0-L3: Volume sonore
M: à1, volume controllé par l'enveloppe

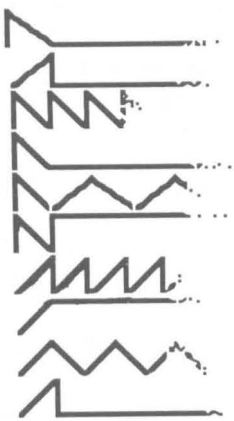
R11 P7 P6 P5 P4 P3 P2 P1 P0 Période de l'enveloppe
R12 P15 P14 P13 P12 P11 P10 P9 P8
P0-P15: Période (à multiplier par 16 pour avoir la période en micro-secondes, soit de 0 à 1 seconde environ)

R13 X X X X Cont ATT ALT Hold Numéro de l'enveloppe
CONT: répétition du motif initial
ATT: front initial (0= descendant, 1= montant)
ALT: alterner front montant/ descendant
HOLD: Continuer le son indéfiniment

R14 IOA7 IOA6 IOA5 IOA4 IOA3 IOA2 IOA1 IOA0 Port A

Figure 8912-C

	CONT	ATT	ALT	HOLD
0 - 3	0	0	X	0
4 - 7	0	1	X	X
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1



C) LES CONNECTEURS

1-Généralités

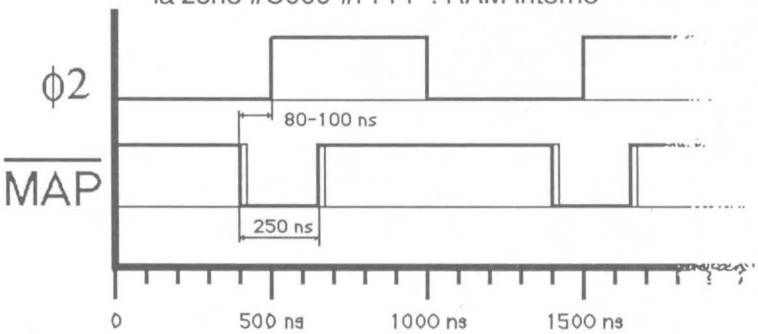
Pour communiquer avec le monde extérieur, l'Oric intègre un certain nombre d'interfaces, qui sont:

- -Un bus complet permétant le branchement de lecteur de disquettes etc...
- -Un connecteur imprimante
- -Une sortie vidéo RVB
- -Une sortie vidéo composite (PAL)
- -Une prise K7 permettant la sauvegarde/lecture de programmes.

Figure connecteur C

2-Le bus complet

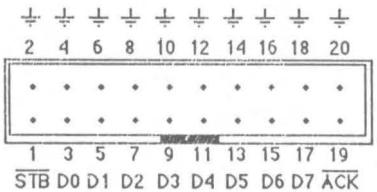
ROMDIS: Déconnecte la ROM s'il est à 0 .
I/O: Tombe à 0 pour une adresse comprise entre #300 et #3FF
I/O control: Déconnecte le VIA interne
MAP: actif à 0 selon le timing ci-dessous. Effet:
la zone #0000-#BFFF : mémoire externe
la zone #C000-#FFFF : RAM interne



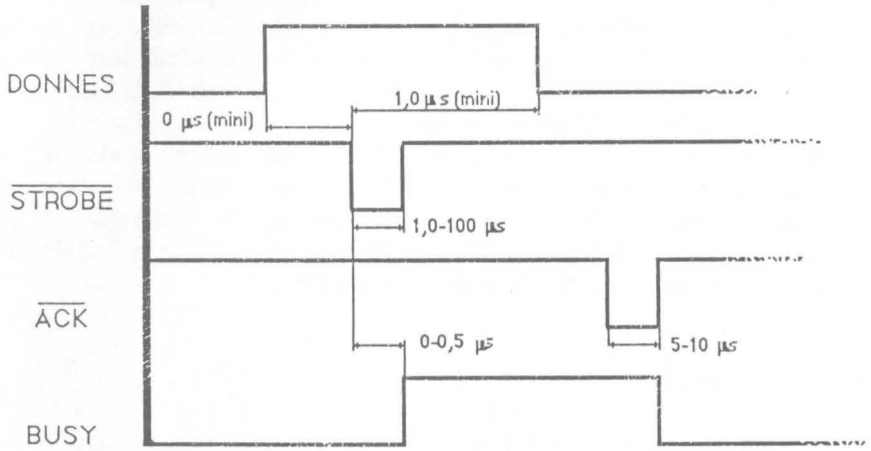
3-Le connecteur imprimante

Figure connecteur B

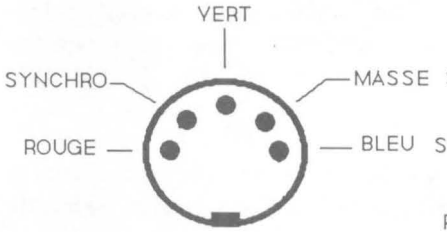
BROCHAGE DU CONNECTEUR IMPRIMANTE



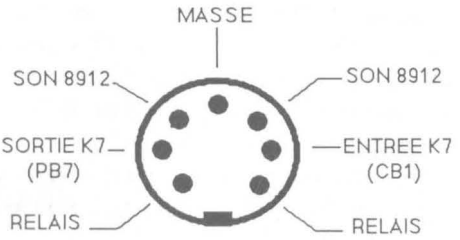
TIMING DE L'ENVOI D'UN CARACTERE



PRISE VIDEO



SORTIE MAGNETO



4-Les connecteurs vidéo et cassette

D) LES INTERFACES UTILISATEURS

1-Généralités

Pour réaliser ses propres applications, on dispose soit du bus complet, soit du port imprimante, ou encore du connecteur cassette. Ce dernier peut être très utile pour des transmissions séries entre Oric's, ou pour des applications de synthèse vocale, fréquence-mètre etc... On peut de même utiliser le relais pour commuter des tensions assez élevées, mais gare aux connecteurs !

2-Le port imprimante

Le port imprimante est souvent suffisant, il dispose de possibilités intéressantes, notament la possibilité de générer des IRQ (grace à CA1). Il ne faut cependant pas oublier que le port est aussi relié au PSG 8912. Pour éviter les interférences, il est recommandé de se servir du signal STROBE pour valider sa propre application, comme celà est utilisé pour l'imprimante. Il est aussi recommandé d'isoler le VIA par des buffers.

Toujours à cause de l'organisation des E/S, il est malaisé d'utiliser le port imprimante en entrée, car le port A du VIA sert aussi à scruter les colonnes du clavier. Il faut donc soit se passer de quelques colonnes, ce qui est malaisé en mode direct, soit isoler le port, et ne le mettre en entrée que pendant les instants strictement nécessaires aux E/S.

3-Le bus complet

Pour des applications plus sérieuses, il faut utiliser le bus complet. Il va alors falloir faire face au problème suivant: à quelle adresse placer ses extensions ?

Le plus simple est d'utiliser la zone #300-#3FF, car un prédécodage (A8-A15=#03) est fait par la broche I/O. Il ne reste plus alors qu'à décoder au maximum A0-A7.

Il faut faire attention à laisser travailler le VIA interne. En effet, celui-ci est sélectionné par I/O, c'est à dire pour #300-#3FF. Comme il occupe 16 registres, on le retrouve 16 fois dans cette zone. Toutefois, la ROM fait exclusivement référence à la zone #300-#30F. Il est donc interdit d'y placer d'autres extensions.

En revanche, il faut impérativement déselectionner le VIA (I/O control) lorsqu'on désire adrésser la zone #310-#3FF, sinon 2 boitiers seraient adréssés en même temps.

Il est de même recommandé d'éviter les zones suivantes:

-#310-#31B, utilisée par le lecteur de disquette Oric

-#31C-#34F, utilisé dans les version futures de l'Oric

-#3F4-#3FF, utilisée par le lecteur de disquette TRAN

Les autres extensions courantes naviguent entre #320 et #3FF. Mieux vaut donc prévoir la possibilité de changer facilement l'adréssage de l'extension.

Il faut remarquer qu'un décodage exhaustif n'est réellement utile que pour des applications commerciales. Un décodage utilisant uniquement IO et A7, sélectionnant pour le périphérique la zone #380-#3FF est parfaitement envisageable.

CHAPITRE III

ORGANISATION LOGICIELLE

A) ORGANISATION DE LA MEMOIRE

1-La carte mémoire

Voici la carte mémoire de l'Oric, selon le mode TEXT/GRAB ou HIRES:

FIGURE III-A

(voir page 28)

2-Les contraintes

Voici quelques contraintes qui ont influencé l'élaboration de la carte mémoire:

-La ROM.

Etant donné que le 6502 place son vecteur de RESET en #FFFC, il est préférable d'implanter la ROM en haut de la mémoire, le vecteur de RESET étant ainsi intégré. Le contraire nécessiterait un décodage et un détournement de l'adresse #FFFC-D lorsque celle-ci se présente sur le bus, ce qui est lourd et inutile ici.

De plus, pour permettre de n'utiliser qu'un seul boitier de 16 Ko, la ROM doit être stockée à des adresses contigües.

C'est donc tout naturellement que la ROM est implantée de #C000 à #FFFF.

-Le 6502.

Le processeur impose ses contraintes, outre le vecteur de RESET vu précédemment.

La pile doit obligatoirement être implantée en #100-#1FF

Le role particulier de la page 0 ne la destine pas à être une simple 'mémoire programme'.

Le 6502 ne peut adresser directement que 64 Ko. Pour pouvoir adresser les 80

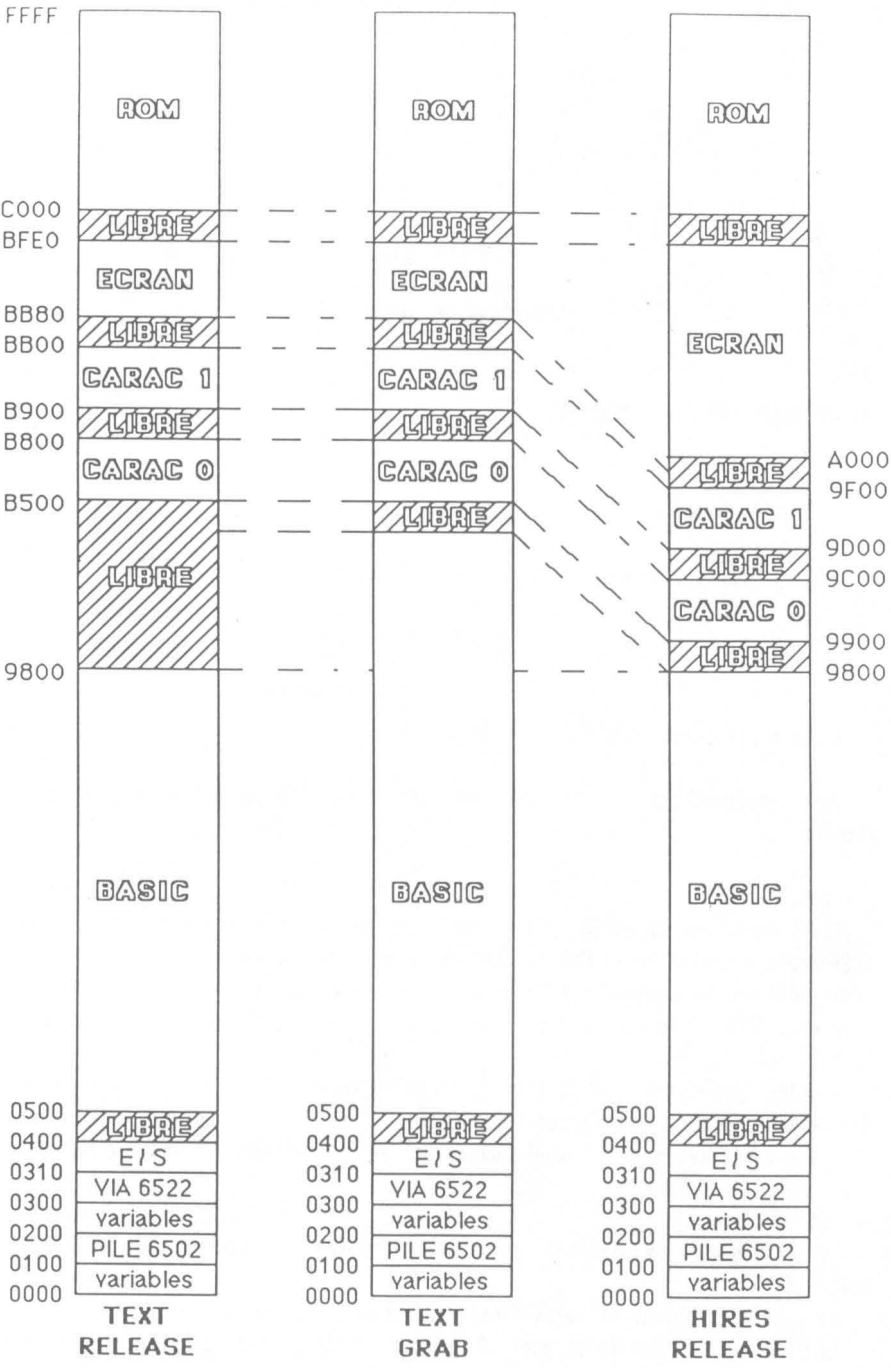

FFFF
ROM
C000
BFE0
LIBRE
ECRAN
BB80
BB00
CARAC 1
B900
B800
CARAC 0
B500
9800
BASIC
0500
0400
E/S
0310
VIA 6522
0300
variables
0200
PILE 6502
0100
0000
TEXT
RELEASE
TEXT
GRAB
HIRES
RELEASE
A000
9F00
9D00
9C00
9900

Figure III-A

Ko (64 Ko de RAM et 16 Ko de ROM), il faut utiliser l'artifice de la pagination: les 16 Ko de #C000 à #FFFF sont accessibles pour la ROM ou la RAM, selon les signaux MAP et ROMDIS. Lorsqu'un lecteur de disquettes est connecté, la zone #E000-#FFFF est même accessible comme EPROM du controleur, ce qui porte à 88 Ko la mémoire adressée par le 6502. Lorsque plusieurs pages mémoires sont accessibles plusieurs fois avec des adresses identiques, on parle de technique d'overlay.

Bien entendu, il faut des routines pour passer de la RAM overlay à la ROM. Ces routines sont généralement implantées dans la zone #400-#4FF.

La RAM overlay aurait aussi bien pu être placée de 0 à #3FFF par exemple. Ceci aurait eu l'avantage de disposer de 2 piles et de 2 pages 0, facilitant ainsi la mise en oeuvre du Dos. Mais cela aurait compliqué les choses: les adresses périphériques seraient adressées 2 fois (#300-#3FF), les vecteurs d'interruption restaient en ROM etc...

-Le Basic.

Le Basic Microsoft utilise presque toute la page 0. Les extensions graphiques et les routines systèmes ayant besoin de mémoire de travail en RAM, il a fallu réserver une autre page mémoire pour stocker les variables systèmes. La page 1 étant prise par la pile, la page 2 a été naturellement choisie.

-Les périphériques.

Le 6502 'voyant' les périphériques comme des adresses mémoire, il faut leur réserver une zone particulière, il est ainsi facile à l'Oric de fournir des signaux facilitant le décodage des adresses par des périphériques externes. La page 2 étant déjà prise, ce sera la page 3 qui sera considérée pour les périphériques.

Notons que cette facilité a effrayé certains puisqu'on a pu voir des interfaces qui décodaient des adresses de la ROM inutilisées (sur la V1.0 seulement) !.

-L'ULA.

Il restait donc la zone #500-#BFFF pour implanter les mémoires d'écrans, puisque le choix d'un processeur vidéo indépendant disposant de sa propre RAM n'a pas été fait, pour des raisons de cout probablement. Il était bien sur nécessaire que la mémoire d'écran soit en un seul bloc, et de préférence pas en plein milieu... Elle se retrouve donc au sommet de la RAM, de #9800 à #BFFF.

Pour faciliter le travail de l'ULA, des adresses 'arrondies' ont été choisies. Beaucoup d'autres contraintes ont du présider à la conception de cette ULA dont on ne sait rien...

Voilà donc le pourquoi de la carte mémoire.

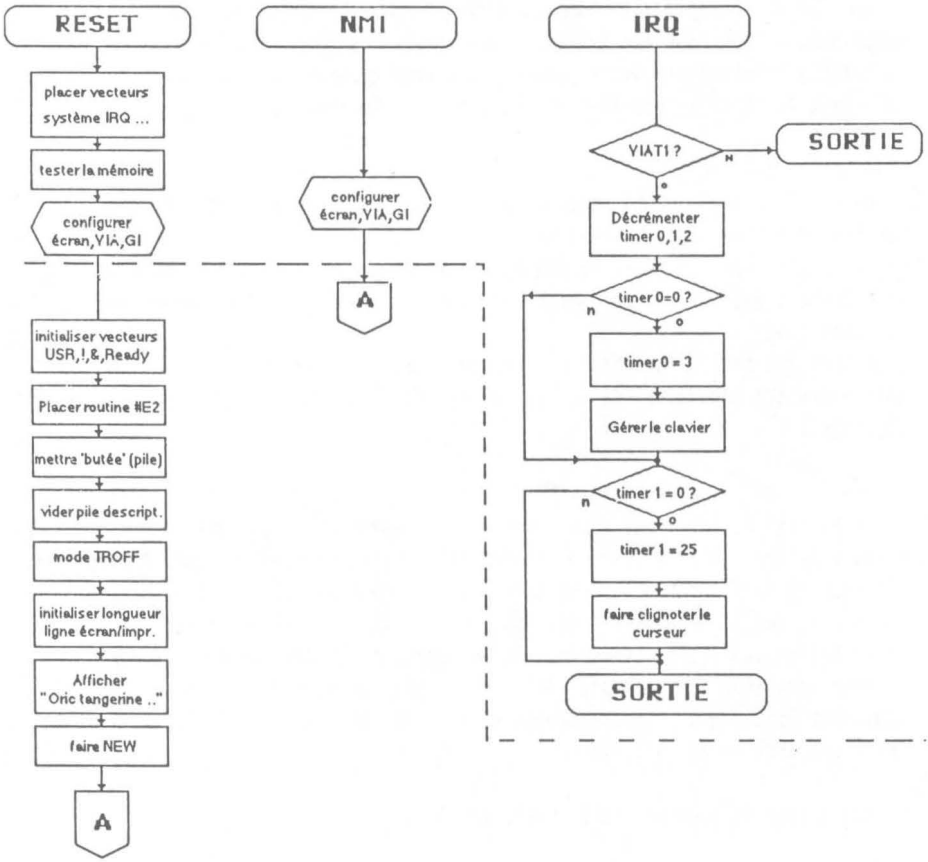
B) UTILISATION DU 6502

1-Principe général

Le fonctionnement global du système est régi par les trois interruptions (RESET, NMI, IRQ) et le programme principal, interrompu éventuellement par les intérruptions.

Voici donc l'organigramme général du travail du 6502: sachant que tout commence par un RESET, et que les IRQ sont générées normalement tous les 1/100 ème de seconde.

Figure III-B



On peut distinguer pour les RESET et NMI un niveau système et un niveau langage. Le niveau système s'occuppe de configurer le VIA 6522, le PSG 8912, l'écran (et l'ULA), alors que le niveau langage s'occupe de placer correctement tout ce qui préside à la bonne marche de l'interpréteur BASIC.

Le 6502, après un RESET ou une NMI, se retrouve à l'interprèteur, c'est à dire qu'il attend ou exécute un ordre.

Périodiquement, il est détourné de sa tache et va gérer le clavier et le clignotement curseur. Ce détournement ayant lieu très souvent (mais pas pour longtemps), on a l'illusion que le 6502 éffectue deux taches simultanément.

Il est même assez facile de faire travailler l'Oric en multi tache, c'est à dire lui faire exécuter 2 programmes Basic 'simultanément'. En fait, tous les 1/40 ème de seconde par exemple, le 6502 passera de l'un à l'autre.

Naturellement, plus le 6502 passe de temps dans les interruptions, moins il passe de temps à exécuter le programme principal.

Notons enfin que le choix qui a été fait de gérer le clavier dans les interruptions est assez douteux, puisque la gestion du clavier prend inutilement 20 % du temps pendant l'exécution d'un programme, alors qu'il suffirait de le gérer uniquement lorsque l'interpréteur en a besoin (GET, KEY$, entrée de commande).

CHAPITRE IV

LE BASIC:UN LANGAGE

A) GENERALITES

1-Les niveaux de langage

Il existe deux niveaux de langage:
-Le langage machine
-Les langages de hauts niveaux

Le langage machine a pour lui une efficacité optimale du programme générée, et contre lui son absence quasi totale de portabilité (autant de langages assembleurs que de microprocesseurs) et un manque flagrant de lisibilité.

L'avenir est donc aux langages de hauts niveaux (BASIC,PASCAL,C etc...), qui, s'ils ne produisent pas des codes optimisés (les capacités de la machine ne peuvent être pleinement exploitées), offrent des programmes lisibles et surtout transportables, du moins en théorie.

2-Langage interprété/compilé

Il est toutefois évident que le microprocesseur ne peut exécuter que du langage assembleur (encore que des processeurs microprogrammés dans certains langages commencent à faire leur apparition).

C'est au moment de la traduction d'un programme écrit en langage évolué en code exécutable par le processeur (ici le 6502) que se distinguent deux types de langages évolués. Il s'agit des langages interprétés et des langages compilés.

Les langages compilés

Les langages compilés traduisent en un programme en langage machine, appelé

programme objet, le programme rédigé en langage évolué, appelé langage source. Le programme est alors exécuté comme s'il s'agissait d'un programme en langage machine, d'où une grande performance.

Le plus simple des langages compilés est l'assembleur lui-même. C'est aussi le plus puissant: le programme est le reflet direct du programme source. Toutefois, l'assembleur n'est pas un langage évolué, puisqu'il n'est pas transportable.

Il existe bien d'autres langages compilés: BASIC, PASCAL, C, FORTH etc...

La performance d'un compilateur se voit à l'optimisation du code généré, qui n'est pas la même selon les langages ou selon les compilateurs d'un même langage.

Prenons un exemple précis, celui d'une simple boucle de 1 à 100.

Figure BAS-A

BASIC:

```
FOR I=1 TO 100
NEXT I
```

COMPILATEUR 1

```
      LDX  #00
LOOP  INX
      CPX  #100
      BNE  LOOP
```

COMPILATEUR 1

```
      LDX  #00
      LDY  #00
LOOP  INX
      BNE  SUITE
      INY
SUITE CPX  #100
      TYA
      SBC  #00
      BCC  LOOP
```

COMPILATEUR 3

```
      LDX  #100
LOOP  DEX
      BNE  LOOP
```

On le voit, la traduction en langage machine est différente, pour un même résultat. Mais certains programmes sont plus courts ou plus rapides que d'autres.

Ce court exemple permet d'entrevoir la difficulté d'écrire un compilateur.

Le compilateur 1 a repéré que l'indice de boucle ne dépassait pas 255, il a pu donc utiliser un seul index.

Le compilateur 2 impose d'office un indice maximum à #FFFF par exemple, il s'ensuit un manque d'optimisation pour les indices inférieurs à 256.

Le compilateur 3 a repéré (ou imposé) que l'indice ne servait qu'à compter (ainsi que le préconisent les gourous de la bonne programmation) , le sens de variation de l'indice n'intervient donc pas, et l'indice a été pris décroissant, ce qui est meilleur avec le 6502.

Les langages interprétés.

Pour les machines peu puissantes, les langages compilés sont peu utilisables, à cause notament du peu de mémoire vive, et de l'absence en général de mémoire de masse rapide. Il faut en effet faire alterner sans cesse en mémoire le texte source et le programme objet, ainsi que l'éditeur de texte et le compilateur.

On a donc recourt à un 2 ème type de langage évolué: les langages interprétés.

Cette fois, le programme reste dans un état 'source', et est transformé en une suite de codes machine seulement à l'exécution, et au coup par coup.

L'exécution d'un ordre nécéssite maintenant 2 opérations: la recherche et la reconnaissance de l'ordre, (phase d'interprétation), et l'exécution du sous programme assembleur correpondant.

Il n'y a plus un seul pointeur de programme (PC) mais deux: le pointeur du programme interprété, et PC.

Voyons comment la simple boucle de tout à l'heure va s'exécuter avec un BASIC interprété.

1) Se placer au début du programme
2) Reconnaitre l'instruction FOR
3) Sauter l'instruction FOR
4) Prendre, et éventuellement créer la variable I
5) Demander un '='
6) Evaluer une expréssion numérique
7) Et donner sa valeur à I
8) Demander TO et sauter
9) Evaluer une expression numérique
10)Sauver la valeur du pas, la valeur limite, le nom de l'indice
11)Ajouter le pas à l'indice
12)Si l'indice est plus grand que la valeur limite,sortir
13)Retour en 11.

Compliqué n'est-ce pas ? Et lent, puisque chaqu'une des opérations nécessite jusqu'à 1 Ko de code machine...

En fait,il n'y a pas de miracle: si on facilite la tache de l'utilisateur, on complique celle de la machine. Et compliquer la tache de la machine, c'est surtout lui faire perdre du temps.

Pour simpifier, on peut dire que le langage interprété fait pendant l'exécution ce qu'un langage compilé fait une seule fois lors de la compilation.

En fait, tous les langages interprétés utilisent un code plus ou moins éloignés du code ASCII, pour éviter du travail lors de la phase d'exécution.

Si le code est déjà très travaillé, on parle alors -abusivement- de langage semi-compilé.

Par exemple, les nombres peuvent être stockés en ASCII ou convertis en flottant dès l'entrée de la ligne, ce qui évite de le faire à chaque fois.

La manière dont le programme est codé sera expliquée plus loin.

En résumé, le langage BASIC de l'ORIC est un langage interprété de haut niveau, lui même rédigé en langage machine, langage de bas niveau.

3-A-t-on affaire à un Basic Microsoft ?

De tout évidence, le BASIC de l'Oric est signé Microsoft, sauf pour les routines sons et graphiques, ainsi que les routines systèmes (IRQ, initialisations...)

D'ailleurs, un copyright Microsoft est caché dans la ROM (#E431/#E435).

La partie principale de la ROM (#C000-#E500 environ) est donc signée Microsoft. Outre le copyright, ceci peut se repérer de deux manières:

-La programmation: le code Microsoft est très optimisé, de la programmation de professionnel à n'en pas douter. Le reste de la ROM, se distingue par une programmation souvent approximative et très peu optimisée. La place gagnée par une meilleure optimisation permettrait de rajouter des commandes supplémentaires.

-Les routines peuvent se retrouver, à l'emplacement près, dans des ordinateurs qui se réclament ouvertement ou non Microsoft: Apple, CBM 64, Vic 20 etc.

En fait, presque tous les Basics 6502 ont un noyau Microsoft, ce qui se comprend compte tenu du temps nécéssaire pour générer un code aussi optimisé, et l'Oric n'échappe pas à la règle.

D'ailleurs, les ROM's de ces ordinateurs ont toujours la même structure: noyau Microsoft, commandes spéciales (graphique pour l'Oric par exemple) et routines systèmes dépendant directement du micro ordinateur utilisé.

B) LE CODAGE D'UN PROGRAMME

1-Le codage du programme

Pour les raisons expliquées précédemment, le programme n'est pas stocké directement en ASCII, mais est déjà un peu codé, de manière à être plus rapidement analysé par la ROM.

Le programme est codé, on s'en doute, comme un ensemble de lignes.
Les lignes sont stockées par ordre de numéro croissant, indépendament de l'ordre d'entrée.

Le codage d'une ligne

Il comprend d'une part des informations 'systèmes' et d'autre part les instructions elles même.

-Une ligne débute par 5 octets qui sont, dans l'ordre:

ØØ LKL LKH LL LH
¥ØØ marque le début de la ligne
¥LKL et LKH (abréviation de LINK) représentent le lien interligne. C'est-à-dire en fait l'adresse de la ligne suivante, ajustée après le Ø de la ligne suivante.

La fin du programme est marquée par LKH=#ØØ
¥LL et LH représente le numéro de la ligne, codé sur deux octets en binaire.

-Il vient ensuite le programme lui-même,pas codé ou presque: seuls les mots-clés du BASIC sont compactés en un seul code dont b7 est forcé à 1. Ils sont donc facilement repérables.
Les mots clés sont reconnus lors de l'entrée de la ligne par une routine spéciale. Le code associé à un mot clé est appelé "token".
Les mots clés ne sont pas tout à fait dans le désordre, on s'en doute. Voici les différentes catégories:

¥de #8Ø à #C1, les commandes
¥de #C2 à #CB, les attributs divers
¥de #CC à #D5, les opérateurs
¥de #D6 à #ED, les fonctions à un seul paramètre
¥de #EE à #F1, les fonctions sans arguments
¥de #F2 à #F3, les fonctions à plusieurs paramètres
¥de #F4 à #F6, les fonctions de segmentation de chaines

Tous le reste de la ligne est stocké en ASCII. Il faut noter que la routine

de codage veille à ne pas coder les chaines entre guillemets, ni les REM ou DATA.

Un codage un peu plus complet aurait beaucoup accéléré le BASIC: coder les constantes numériques en flottant, les numéros de lignes après GOTO sur deux octets, etc... Cette 'semi compilation' permet à certains BASIC interprétés d'afficher des performances étonnantes. Il est vrai que la ROM se trouve largement rallongée (24 à 32 Ko en général)

BASIC2.TXT

Le codage des variables

L'avantage du BASIC face à d'autres langages de haut niveau, c'est sa gestion dynamique des variables, qui est très simle d'emploi, et ne s'embarrasse donc pas de diverses déclarations etc...

Les variables sont stockées dans une zone à part, après le texte BASIC, et au maximum jusqu'au plafond fixé par le HIMEM

Cet espace est divisé en trois zones:

-Les variables simples
-Les tableaux
-Les chaines.

Les variables simples

Elles sont stockées sur 7 octets. La structure est la suivante:

L1 L2 D0 D1 D2 D3 D4

*L1 et L2 représentant les deux caractères significatifs de la variable. Ils servent aussi à distinguer de quel type de variable il s'agit: b7 est parfois forcé à 1. Si la variable n'a qu'un caractère, le 2 ème caractère est considéré comme valant 0.

Voici les codages selon le type:

type	L1	L2
Réel	b7=0	b7=0
Entier	b7=1	b7=1
Chaine	b7=0	b7=1
Fonction	b7=1	b7=0

*D0 à D4 représentent la valeur de la variable. Bien entendu, la signification est différente selon le type.

Réel: D0=exposant
D0-D4=mantisse. Le signe est incorporé dans b7 de D1.

Entier: D0=poids fort
D1=poids faible
D2-D4=0.

Chaine: D0=longueur (donc 0 si la chaine est vide). La longueur maxi est de #FF, soit 255 et non 256.
D1-D2=adresse de la chaine.
D3-D4=0

Fonction: D0-D1=adresse de la définition
D2-D3=adresse de la variable
D4=adresse du premier octet de la définition (inutilisé, voir routine DEF FN)

Etant placées en premier, les variables ont une caractéristique très importante: leur adresse ne varie plus une fois leur place créee. C'est important pour les boucles FOR-NEXT, qui peuvent ainsi stocker l'adresse de la variable et non son nom, évitant ainsi une recherche à chaque NEXT.

Les tableaux

Les tableaux ont une structure plus complexe. Ils sont stockés après les variables, et leur adresse peut donc changer après une nouvelle allocation de variable.

Voici la structure d'un tableau:

L1 L2 TL TH ND DnH DnL ... D0H D0L Données...

#L1 L2: c'est le nom du tableau, avec les mêmes conventions que pour les variables simples. Les fonctions ne sont pas admises.
#TL TH: c'est la longueur totale du tableau (entête compris). Ceci permet de sauter facilement d'un tableau à l'autre lors de la recherche (c'était inutile pour les variables, puisqu'une variable simple occupe toujours 7 octets).
#ND: c'est le nombre de dimensions du tableau
#DnH DnL ... D0H D0L: (attention, poids fort d'abord) ce sont les composantes dans chaque dimension. La valeur pour la dernière dimension est stockée en premier à cause de la structure de la routine d'évaluation d'un tableau.
#Données: c'est (enfin !) le contenu du tableau. Cette fois, la place n'est pas gaspillée, et la place strictement nécéssaire est utilisée: 5 octets pour un réel, 2 octets pour un entier et 3 octets pour une chaine. Ces octets ont la signification décrite pour les variables simples.

Evidemment, les données sont stockées dans un ordre logique. Leur rang, en fonction des composantes, peut être donné par la formule suivante:

Convention: Le tableau a été dimensionné par:

DIM A (C_0-1, C_1-1, ... , C_n-1)

et l'élément est repéré par :

A (V_0, V_1, ... , V_n)

Le rang de l'élément est alors donné par l'expression:

$R=((((V_n*C_{n-1}+V_{n-1})*C_{n-2}+V_{n-2})$ .... $)*C_1+V_1)*C_0+V_0$)

C'est ainsi qu'est calculée l'expression dans la ROM . En développant, on obtient la formule suivante (Π=produit, Σ=somme)

$$R=\sum_{i=0}^{n} V_i \prod_{\substack{j=0 \\ i<>0}}^{i-1} C_j$$

Par exemple, pour DIM A(2,2,2)
on a $C_0=C_1=C_2=3$
et le rang est donné par la formule suivante:

Figure IV-A

$(V_2C_1+V_1)C_0+V_0$ ou $V_0+V_1C_0+V_2C_0C_1$
ou encore $(3V_2+V_1)3+V_0$ ou $V_0+3V_1+9V_2$

Ce qui donne, pour un tableau dimensionné par DIM A(2,2,2) par exemple, les éléments dans l'ordre suivant:

```
rang  0: 0,0,0 1,0,0 2,0,0 0,1,0 1,1,0 2,1,0
rang  6: 0,2,0 1,2,0 2,2,0 0,0,1 1,0,1 2,0,1
rang 12: 0,1,1 1,1,1 2,1,1 0,2,1 1,2,1 2,2,1
rang 18: 0,0,2 1,0,2 2,0,2 0,1,2 1,1,2 2,1,2
rang 24: 0,2,2 1,2,2 2,2,2
```

Pour obtenir l'adresse d'un élément, il faut donc calculer son rang, le multiplier par la longueur d'un élément (2,3 ou 5 selon le type) et ajouter à l'adresse de début des données. C'est exactement dans cet ordre qu'agit la routine qui recherche un élément d'un tableau.

C) OPTIMISATION DES PROGRAMMES BASIC

1-Compacter un programme

Nous allons passer en revue quelques trucs simples, tirés du fonctionnement de l'interpréteur, qui permettront d'accelérer ou de compacter un programme BASIC, ou parfois les deux en même temps...

-Ce qui vient tout de suite à l'esprit, c'est de supprimer les REM et les espaces inutiles.
On pourra écrire une petite routine, facile si vous avez bien assimilé les routines REM et DATA.
-Si des expressions arithmétiques se rencontrent trop souvent, utiliser la définition des fonctions FN. Utiliser de même des sous programmes pour des séquences répétitives.
-Mettre le plus d'instructions possible par ligne (une ligne gaspille 5 octets) . Une ligne, pour être traitée correctement, peut avoir une longueur quelconque, à quelques exeptions près: les REM, DATA, et les chaines de caractère ne doivent pas dépasser 256 caractères.
Rien n'empêche donc d'écrire un programme qui compacte, en gardant à l'esprit que le programme compacté ne sera plus éditable, et en faisant attention aux GOTO et GOSUB, qui doivent toujours adresser un début de ligne.
-Si des constantes sont souvent utilisées, il est préférable de les affecter à des variables.
-Avec l'aide d'un renuméroteur, on aura avantage à diminuer les numéros de ligne: numéroter à partir de 1000 au lieu de 10000 fait gagner un octet à chaque branchement.
-Eviter les affectations inutiles: A$="#"+A$: POKE I,VAL(A$) est équivalent à POKE I,VAL("#"+A$). C'est aussi bénéfique pour la rapidité.

2-Accélérer un programme

-Eviter des ligne 'mortes', que l'interpréteur doit sauter lorsqu'il passe dessus, et qui augmente le temps de recherche d'une ligne par GOTO ou GOSUB. Eliminer donc les REM, et rejeter les DATA à la fin d'un programme. On peut estimer qu'il faut 1 ms pour 'passer dessus' une instruction DATA ou REM.
-Mettre le plus possible d'instructions par ligne, puisque l'interpréteur perd un peu de temps à prendre les informations de début de ligne. Ce temps est d'environ de 0,1 ms.
-Eviter les espaces inutiles. L'interpréteur perd 17 microsecondes par espace inutile.

Voilà pour la 'forme' même du programme. Nous allons voir maintenant

comment faciliter la tache du BASIC: moins il en a à faire, plus il en fait !

-Le point le plus sensible consiste à éviter le plus possible les évaluations numériques ou de chaines, et en premier lieu les conversions décimal flottant. On préfèrera donc affecter les constantes à des variables.

La différence est énorme: A=1000 est 4 ms plus lent que A=MI, si MI vaut 1000.

Beaucoup d'instructions passent tout leur temps à évaluer les paramètres, par exemple POKE, etc... Il convient donc de leur faciliter la tache.

L'évaluation d'un nombre hexadécimal est beaucoup plus rapide que celle d'un nombre décimal. Par exemple, 900 nécéssite 2 ms de plus que &384 pour être évalué ! Ainsi, la plupart du temps, les GOTO passent plus de temps à évaluer le numéro qu'à rechercher la ligne. Si on écrivait les numéros de ligne en hexa, chaque branchement serait environ deux fois plus rapide ! Le programme serait totalement illisible, il est vrai.

-Toujours pour accélérer les branchements, numéroter le programme avec des petits numéros de lignes, pour faciliter l'évaluation des numéros.

-Ne jamais utiliser des variables entières, car les routines ne calculent qu'en flottant, ce qui implique une conversion entier flottant pour le calcul, suivie d'une conversion flottant/entier. On peut estimer que l'on perd environ 1 ms à chaque appel à une variable entière, par rapport à un variable normale.

-Réduire les noms des variables les plus courantes à une seule lettre, et les déclarer le plus tot possible pour accélérrer leur recherche (elles seront ainsi stockées au début de la table des variables).

-Eviter l'emploi de tableaux, si c'est possible, ou en tous les cas réduire le nombre de dimensiosn, car le calcul de l'adresse d'un élément prend du temps (une multiplication et une addition par composante), sans compter qu'il y a autant d'évaluations de variables que de dimensions.

-En revanche, il est souvent plus rapide de mettre en tableau des données que de calculer une expréssion à chaque fois.

-Ne pas mettre de variables après les NEXT.

-Préférer les structures FOR-NEXT aux structures REPEAT-UNTIL, car ces dernières nécéssitent l'évaluation d'expréssions pour le test de sortie.

-Simplifier les expressions numériques: supprimer les parenthèses inutiles, factoriser les expressions pour faire le moins de multiplications possible.

-Ne pas oublier que les commandes IF A()0 et IF A sont totalement équivalentes (voir commande IF).

-Enlever ou ralentir la scrutation clavier, ce qui augmente de 20 % la vitesse d'un programme quelconque. (POKE &30E,&40 pour enlever et POKE &30E,&C0 pour autoriser, et ne pas oublier le décimal pour la V1.0 !)

-Quelques remarques d'ordre général, assez indépendantes du langage:

-Optimiser au maximum les boucles exécutées souvent: il n'est pas très utile d'optimiser des routines exécutées une seule fois.

-Eviter l'appel des sous programmes. Le programme sera plus long mais plus rapide.

-Un meilleur algorithme permet d'aller plus vite, sans artifice de programmation. Notamment, éviter au maximum les calculs transcendants, qui sont très longs. Voir l'exemple de la commande CIRCLE.

Ces différentes remarques permettent de doubler facilement la vitesse d'exécution d'un programme BASIC. Il est toutefois conseillé de garder une version non modifiée du programme. Enfin, si c'est toujours trop lent, tournez-vous vers l'assembleur !

CHAPITRE V

LES VARIABLES SYSTEMES

A) INTRODUCTION

1-Introduction

Le terme de variable système est en fait abusif, il recouvre ici deux notions: les variables de travail, qui sont divers pointeurs, accumulateurs flottants, et des données systèmes qui influencent le comportement de la ROM, ce sont par exemple les vecteurs de traitement des IRQ, les adresses de l'écran etc...

Les vraies variables systèmes ne sont touchées que par des commandes de configuration, et notamment initialisation. Elles peuvent se trouver aussi bien en RAM qu'en ROM. Dans ce cas, on pourra plutot les appeler données systèmes, ou constantes.

Il est parfois difficile de dire si ce sont des variables de travail, ou des données systèmes. En fait vont être exposés dans ce chapitre la signification des trois premières pages de la RAM: la page Ø,la Page 1,la page 2.

B) LA PAGE Ø

1-Role pour le 65Ø2

La page zéro (adresses de ØØØØ à ØØFF) joue un role particulier pour le 65Ø2, certaines instructions y faisant explicitement référence.

Ainsi, le fait que le Basic soit implanté en ROM interdit au programme de s'autotransformer (ce qui n'est jamais avantageux de toutes façons). Pour travailler sur la mémoire, il faut donc utiliser l'indirect, qui nécéssite un pointeur en page Ø, qui est le seul mode autorisant un accès indirect. (seul le JMP indirect peut se servir d'un pointeur quelconque).

Par la constitution même du 6502, la page 0 a d'autres avantages la lecture ou l'écriture d'un octet s'obtient avec seulement un octet d'adresse, le poids fort étant forcé à 0. Ce qui réduit d'un octet chaque appel à une telle variable, et permet de gagner un cycle machine, celui qui aurait servi à aller chercher le poids fort de l'adresse.

A ce sujet, il convient de se méfier: une instruction en page 0, même indexée, ne touchera jamais au poids fort de l'adresse (sauf pour l'indirect évidemment). Ainsi LDA #FF,X si X vaut 2 est équivalent à LDA #01 et non LDA #101.

C'est la page 0 qui permet les modes d'adressages les plus riches, et la meilleure optimisation d'exécution. Elle a donc été choisie pour toutes les variables de l'interpréteur. La partie non Microsoft y fait rarement appel, ce qui est dommage: les routines haute résolution auraient été beaucoup plus rapides avec des pointeurs en page 0.

2-La page 0 octet par octet

00 Inutilisé
01 Inutilisé
02 Inutilisé
03 Inutilisé
04 Inutilisé
04 Inutilisé
06 Inutilisé
07 Inutilisé
08 Inutilisé
09 Inutilisé
0A Inutilisé
0B Inutilisé

0C Travail
0D Travail
0E Travail
0F Travail

10-11:Adresse de la ligne du curseur Hires

12-13:Adresse de la ligne du curseur Text.

14-15:Adresse des données pour sons préprogrammés.

16 Inutilisé

17 :Indicateur de sortie pour INPUT.

```
18-19:Travail encodage/décodage mots clés

1A   :#4C:JMP #CBED/#CCB0
1B-1C:#CBED/#CCB0

1D-1E:Compte le nombre de lignes rencontrés lors de la recherche d'un numéro
      Adresse pour DOKE

1F-20:Calcul d'adresse de ligne écran

21   #4C:JMP #xxxx vecteur USR
22-23:Adresse de USR. Normalement #D2A0/D336

24   :Terminateur (évaluation de chaines etc..), notamment

25   :Terminateur sauvegarde,notamment

26   :Entrée d'une ligne,contient sa longueur, notamment

27   :Sauvegarde routine d'affichage, drapeau pour la recherche d'une variable

28   :Drapeau chaine:#FF (b7=1) si chaine,#00 (b7=0) si numérique

29   :Drapeau entier:b7=1 si entier,0 sinon

2A   :Drapeau encodage des mots clés, réorganisation.

2B   :Drapeau d'autorisation des variables entières/tableaux.
      Remis à 0 par CLEAR

2C   :Code pour INPUT/GET/READ

2D   :Code pour )=(

2E   :b7=1, clavier inactif. b7=0,clavier actif

2F   :Travail conversion flottant/Hexa

30   :Position du curseur sur la ligne (écran ou imprimante).
      V1.0:  contient la valeur + 13

31   :Longueur d'une ligne d'impression

32   :Position maxi pour la tabulation par ,.
```

33-34:Tampon pour nombres sur deux octets.
Attention:Aucune fonction (y compris USR et &() ne doivent y toucher.

35-84:Tampon clavier

35...:V1.Ø:Nom du programme demandé

49...:V1.Ø:Nom trouvé

5E :V1.Ø:Entête,inutilisé
5F :V1.Ø:Entête,début poids faible
6Ø :V1.Ø:Entête,début poids fort
61 :V1.Ø:Entête,fin poids faible
62 :V1.Ø:Entête,fin poids fort
63 :V1.Ø:Entête,type (#ØØ basic,#8Ø,bloc mémoire)
64 :V1.Ø:Entête,AUTO (si non nul)
65 :V1.Ø:Entête,inutilisé
66 :V1.Ø:Entête,inutilisé

85 :Pointeur de la pile des descripteurs

86-87:Précédent pointeur de la pile des descripteurs
Remarque:#87 est mis à Ø à l'initialisation

88-9Ø:Pile des descripteurs (3 fois trois octets)

91-92:Pointeur de travail...

93-94:Pointeur de travail...

95 :ACC3 octet 1
96 :ACC3 octet 2
97 :ACC3 octet 3
98 :ACC3 octet 4

97-98:Calcul adresse dans un tableau

99 :Inutilisé
9A-9B:Début du texte basic,normalement #5Ø1.Pointe sur le premier lien de ligne

9C-9D:Premier octet des variables,dernier octet du texte Basic.

9E-9F:Fin des variables,pointe sur le premier octet des tableaux

A0-A1:Pointe sur l'octet qui suit la fin du dernier tableau

A2-A3:Plafond des chaines,plus basse chaine. Lors d'un CLEAR,rejoint le HIMEM

A4-A5:Adresse de la chaine réservée et travail divers

A6-A7:HIMEM. Les chaines seront stockées juste en dessous

A8-A9:Numéro de la ligne courante. En mode direct, #A9 contient #FF.

AA-AB:Sauvegarde du numéro de ligne, pour la reprise par CONT

AC-AD:Sauvegarde de la dernière valeur de TXTPTR, pour la reprise par CONT.
Si #AD=0 (mode direct), reprise interdite.

AE-AF:READ:Sauver numéro de ligne courante

B0-B1:Pointeur de DATA

B2-B3:Pointeur de DATA (travail)

B4-B5:Caractères significatifs d'une variable

B6-B7:Adresse d'une variable

B8-B9:Stockage de l'adresse d'une variable.

BA :Evaluer une expréssion, indique la fin de l'expréssion

BB Inutilisé

BC :Evaluer une expréssion, code pour)=(

BD-BE:Travail évaluation FN

BF-C0:Travail chaines...

C1 Inutilisé

C2 :Longueur d'un élément (réorganisation). Vaut normalement 0

C3 :#4C JMP $xxxx:évaluer une fonction
C4-C5:Adresse de la fonction à exécuter

```
C5   :Travail opérateurs

C6   :ACC4,exposant
C7   :ACC4,octet 1
C8   :ACC4,octet 2
C9   :ACC4,octet 3
CA   :ACC4,octet 4

C7-C8:Décalage de bloc:Adresse cible

C9-CA:Décalage de Bloc:Dernier octet à décaler

CB   :ACC5,exposant
CC   :ACC5,octet 1
CD   :ACC5,octet 2
CE   :ACC5,octet 3
CF   :ACC5,octet 4

CE-CF:Décalage de bloc: premier octet à décaler
      Recherche de lignes: adresse de la ligne

D0   :ACC1,exposant
D1   :ACC1,octet 1
D2   :ACC1,octet 2
D3   :ACC1,octet 3   ou adresse descripteur poids faible
D4   :ACC1,octet 4   ou adresse descripteur poids fort
D5   :ACC1,signe

D6   :Travail évaluation nombre décimal

D7   :Justification de ACC1. Doit contenir 0

D8   :ACC2,exposant
D9   :ACC2,octet 1
DA   :ACC2,octet 2
DB   :ACC2,octet 3
DC   :ACC2,octet 4
DD   :ACC2,signe

DE   :Produit des signes de ACC1 et ACC2

DF   :Extension ACC1

DE-DF:Pointeur pour transfert des chaines
```

E0-E1:Sauvegarde de TXTPTR

Routine de lecture d'un caractère

Entrée:Par #00E8 ou par #00E2 si incrémentation.

Sortie:A contient le code du caractère lu,les espaces étant sautés.
Z=1 si fin d'instruction (fin de ligne ou ':')
C=0 si caractère chiffre (0 à 9)
Y et X inchangés

Bogue:Sur la V1.0, X et Y sont modifiés et A contient n'importe quoi si un ELSE ou une REM abrégée (''') sont rencontrés.
(Cf #EA41)

```
E2   :INC E9
E4   :BNE 00E8
E6   :INC EA
E8   :LDA xxxx
EB   :CMP #' '
ED   :BEQ 00E2
EF   :JSR #EA41/#ECB9
F2   :RTS
```

F3 Inutilisé
F4 Inutilisé
F5 Inutilisé
F6 Inutilisé
F7 Inutilisé
F8 Inutilisé
F9 Inutilisé

FA :#80:Valeur courante de RND: exposant
FB :4F:idem,octet 1
FC :C7:idem,octet 2 (première valeur=0,811635196)
FD :52:idem,octet 3
FE :58:idem,octet 4

FF :Début tampon décimal

C) LA PAGE 1

1-Role de pile 6502

La page 1 (#0100-#01FF) a un role très particulier pour le 6502: celui de pile système.

Cette pile sert donc à stocker les adresses de retour des sous programmes, ou les octets à sauver lors des interruptions. Rappelons que la pile a une structure de type LIFO (Last In, First Out: dernier entré, premier sorti). Cette utilisation est classique, nous ne nous y attarderons pas outre mesure.

Le principal problème, c'est que le Basic a aussi besoin d'une pile LIFO, et ce pour les mêmes raisons que le 6502: stocker le moyen de retourner d'un sous programme, gérer des FOR-NEXT imbriqués etc... Et là où le bat blesse, c'est que ces données n'ont absolument pas la même structure que celles stockées par le 6502.

Il serait insensé d'empiler TXTPTR, pointeur du programme interprété, et de le dépiler dans PC, pointeur du programme 6502.

Pour résoudre ces conflits, certains processeurs proposent une pile utilisateur et une pile système. Ce n'est pas le cas du 6502.

Les données basic s'intercalent donc entre les données systèmes, et réciproquement.

Un mot sur le comportement de la pile: il est régi par le pointeur de pile (S pour Stack pointer). Celui-ci descend si on empile une donnée et monte si on dépile. C'est pourquoi il est initialisé au sommet de la pile (#FF,soit #01FF). Signalons à ce sujet le même 'effet modulo' qu'en page 0: si le pointeur vaut #00 (#100) et qu'on empile une donnée, il passera à #FF (#01FF).

Le pointeur indexe toujours la position libre au sommet de la pile. Ainsi le programme suivant récupère la valeur au sommet de la pile sans y toucher:

```
TSX
LDA 0101,X
```

Ce procédé est très utilisé dans la ROM.

Attention: on ne peut espérér récupérer une valeur au delà du sommet de la pile, car les interuptions modifient constamment le sommet de la pile, et ce de maniére totalement transparente pour l'utilisateur.

2-La pile Basic

Elle a deux usages: l'un temporaire, et l'autre permanent.

L'usage temporaire, c'est par exemple lors de l'évaluation d'une expression (voir cette routine). A la fin de la routine d'évaluation, tout ce qui avait

été empilé est a été dépilé, c'est donc transparent pour la pile.

L'usage permanent: c'est la gestion des structures de controle: FOR-NEXT, GOSUB-RETURN, REPEAT-UNTIL.

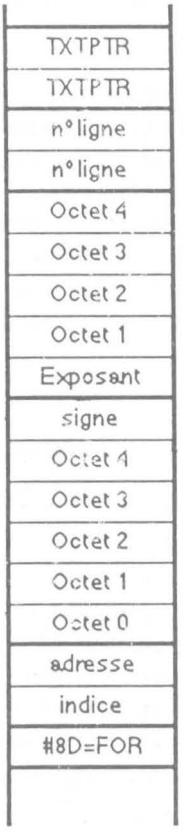
Ces données ont une structure assez similaire. Surtout, le premier octet contient le code du bloc, qui est en fait le token de l'instruction: #8D pour un bloc FOR-NEXT, #9B pour un bloc GOSUB-RETURN, #8B pour un bloc REPEAT-UNTIL.

Le bloc initial (rien donc) est repéré par un #ØØ qui est placé, lors du RESET BASIC, au sommet de la pile (#1FF), et n'est jamais touché. Ainsi, la routine de recheche des blocs (#C3CA/#C3C6) sort lorsqu'elle rencontre le sommet de la pile.

Voici les données empilées pour ces différents blocs:

Figure V-A

CODE: #9B POUR GOSUB
#8B POUR REPEAT
Valeurs empilées pour
GOSUB ET REPEAT



VALEURS EMPILEES POUR FOR-NEXT

Il faut noter qu'une gestion rigoureuse de la pile fait que pratiquement aucune adresse de retour n'est empilée, de sorte que seuls quelques octets au sommet servent de pile système, le reste étant uniquement occupé par les données BASIC.

D'autres structures de pile seront données dans le coeur même du commentaire.

3-Le tampon décimal.

Lors des conversions flottant/décimal ou flottant/hexadécimal, il fallait une zone pour stocker les résultats. Cette zone se situe au début de la page 1, de #1ØØ à #1ØF ou de #ØFF à #1ØE pour l'instruction STR$.

Le nombre est stocké en ASCII, tel qu'il apparait pour un PRINT par exemple, et terminé par un #ØØ.

L'existence d'un tampon décimal, ou de procédures qui peuvent s'imbriquer oblige à tester que la pile ne descend pas trop bas, afin qu'elle ne se sature pas.

Ainsi, chaque fois que le Basic doit empiler quelque chose de manière permanente, ou répétitive, il va tester la place disponible. En pratique, il est testé que la pile de descende pas en dessous de #13E (routine #C43B/#C437), ce qui laisse une marge d'environ #2E (4Ø octets) pour gérer les IRQ et les quelques adresses systèmes.

Il est inutile, à cause de cette marge de sécurité importante, de réserver un nombre précis d'octets sur la pile: si on demande 4 octets disponibles et qu'on en empile 2Ø, il y aura certes 16 octets de dépassement, mais l'erreur sera détectée à la prochaine vérification.

D) VARIABLES SYSTEMES PAGE 2

1-Introduction

Si la page Ø est réservée presque exclusivement à l'interpréteur, c'est à dire à la partie Microsoft du Basic, la page 2 est plutot réservée aux utilisations systèmes (vecteurs d'interruptions...) ou pour les routines spécifiques à l'Oric (Graphiques, son, E/S cassettes pour la V1.1).

Sa structure est moins homogène, les variables ont été allouées au petit bonheur, sans logique apparente. Il en résulte de nombreux trous, et beaucoup de différences entre la V1.Ø et la V1.1.

2-La page 2 octet par octet

200 1.0/1.1 Travail
201 1.0/1.1 Travail
202 1.0/1.1 Travail
203 1.0/1.1 Travail
204 1.0 Travail
1.1 Inutilisé
205 1.0 Travail
1.1 Inutilisé
206 1.0 Travail décalage
1.1 Inutilisé
207 1.0 Travail décalage
1.1 Inutilisé

208 1.0/1.1 Motif ligne/colonne dernière touche pressée

209 1.0/1.1 idem pour touches de controles (Shift,Ctrl,Funct)

20A 1.0/1.1 Motif colonne dernière touche pressée

20B 1.0/1.1 idem pour touches de controle

20C 1.0/1.1 #FF:Majuscule,#7F:Minuscule

20D 1.0/1.1 Travail,compteur des colonnes

20E 1.0/1.1 Compteur de répétition clavier

20F 1.0 Travail accès GI 8912
1.1 Inutilisé

210 1.0/1.1 Travail gestion clavier
211 1.0/1.1 Travail gestion clavier

212 1.0/1.1 FB code

213 1.0/1.1 Registre Pattern

214 1.0/1.1 Travail Pattern

215 1.0/1.1 Motif du sextet du curseur Hires

```
216  1.Ø/1.1 sauvegarde poids faible adresse du curseur (#1Ø)
217  1.Ø/1.1 idem poids fort

218  1.Ø/1.1 sauvegarde motif courant (#218)

219  1.Ø/1.1 Coordonnée horizontale curseur Hires

21A  1.Ø/1.1 Coordonnée verticale curseur Hires

21B  1.Ø/1.1 Circle:Coordonnée relative horizontale,partie fractionnaire
21C  1.Ø/1.1 idem partie entière

21D  1.Ø/1.1 Circle:Coordonnée relative verticale,partie fractionnaire
21E  1.Ø/1.1 idem partie entière

21F  1.Ø/1.1 Ø si mode text,1 en mode Hires

22Ø  1.Ø/1.1 Ø=48K,1=16K (?)

221  1.Ø/1.1
222  1.Ø/1.1
223  1.Ø/1.1
224  1.Ø/1.1
225  1.Ø/1.1
226  1.Ø/1.1
227  1.Ø/1.1
228  1.Ø     #4C:JMP #ECØ3 vecteur IRQ
         1.1 Inutilisé
229  1.Ø     #Ø3
         1.1 Inutilisé
22A  1.Ø     #EC
         1.1 Inutilisé

22B  1.Ø     #4C:JMP #F43Ø vecteur NMI
         1.1 Inutilisé
22C  1.Ø     #3Ø
         1.1 Inutilisé
22D  1.Ø     #F4
         1.1 Inutilisé

22E  1.Ø/1.1 Inutilisé
22F  1.Ø/1/1 Inutilisé
23Ø  1.Ø     #4Ø:RTI  retour normal de l'IRQ
         1.1 Inutilisé
```

```
231  1.Ø/1.1 Inutilisé
232  1.Ø/1.1 Inutilisé
233  1.Ø/1.1 Inutilisé
234  1.Ø/1.1 Inutilisé
235  1.Ø/1.1 Inutilisé
236  1.Ø/1.1 Inutilisé
237  1.Ø/1.1 Inutilisé


238      1.1 #4C:JMP #F77C afficher un caractère
     1.Ø     Inutilisé
239      1.1 #7C
     1.Ø     Inutilisé
23A      1.1 #F7
     1.Ø     Inutilisé

23B      1.1 #4C:JMP #EB78 prendre un caractère au clavier
     1.Ø     Inutilisé
23C      1.1 #78
     1.Ø     Inutilisé
23D      1.1 #EB
     1.Ø     Inutilisé

23E      1.1 #4C:JMP #F5C1 Vecteur imprimante
     1.Ø     Inutilisé
23F      1.1 #C1
     1.Ø     Inutilisé
24Ø      1.1 #F5
     1.Ø     Inutilisé

241      1.1 #4C:JMP #F865 afficher ligne Ø
     1.Ø     Inutilisé
242      1.1 #65
     1.Ø     Inutilisé
243      1.1 #F8
     1.Ø     Inutilisé

244      1.1 #4C:JMP #EE22 vecteur IRQ
     1.Ø     Inutilisé
245      1.1 #22
     1.Ø     Inutilisé
246      1.1 #EE
     1.Ø     Inutilisé
```

247 1.1 #4C:JMP #F8B2 vecteur NMI
1.Ø Inutilisé
248 1.1 #B2
1.Ø Inutilisé
249 1.1 #F8
1.Ø Inutilisé

24A 1.1 #4Ø:RTI retour IRQ normale
1.Ø Inutilisé

24B 1.Ø/1.1 Inutilisé
24C 1.Ø/1.1 Inutilisé
24D 1.Ø/1.1 Inutilisé
24E 1.1 #1Ø:Répétition lente
1.Ø Inutilisé

24F 1.1 #Ø4:Répétition rapide
1.Ø Inutilisé

25Ø 1.Ø/1.1 Inutilisé
251 1.1 Travail affichage:sauver autorisation curseur
1.Ø Inutilisé

252 1.1 b7=1 si un IF a été rencontré (le ELSE doit être donc exécuté)
1.Ø Inutilisé

253 1.1 contient le numéro de la première colonne valide (Ø ou 2)
1.Ø Inutilisé

254 1.Ø/1.1 Inutilisé
255 1.Ø/1.1 Inutilisé
256 1.1 Longueur d'une ligne imprimante
1.Ø Inutilisé

257 1.1 Longueur d'une ligne d'écran
1.Ø Inutilisé

258 1.1 Sauvegarde position horizontale imprimante (#3Ø)
1.Ø Inutilisé

259 1.1 Sauvegarde position horizontale écran (#3Ø)
1.Ø Inutilisé

```
25A     1.1 Drapeau Merge
    1.Ø     Inutilisé

25B     1.1 Drapeau verify/load
    1.Ø     Inutilisé

25C     1.1 Compteur d'erreurs, poids faible
    1.Ø     Inutilisé
25D     1.1 Compteur d'erreurs, poids fort
    1.Ø     Inutilisé

25E     1.1 type de programme (B,C,I,S,R)
    1.Ø     Inutilisé
25F     1.1 ØØØ:fin de la chaine 'type de programme'
    1.Ø     Inutilisé

26Ø     1.1 si 1,mémoire défaillante
    1.Ø     Inutilisé

261 1.Ø/1.1 Poids faible adresse de gestion des caractères de controle
262 1.Ø/1.1 Poids fort idem

263 1.Ø/1.1 Travail calcul adresse d'une ligne
264 1.Ø/1.1 idem poids fort

265 1.Ø/1.1 Travail affichage curseur

266 1.Ø/1.1 Inutilisé
267 1.Ø/1.1 Inutilisé
268 1.Ø/1.1 numéro de la ligne du curseur Text

269 1.Ø/1.1 colonne curseur Text

26A 1.Ø/1.1 Registre d'état

26B 1.Ø/1.1 Valeur colonne Ø (PAPER)

26C 1.Ø/1.1 Valeur colonne 1 (INK)

26D 1.Ø     adresse de l'écran
        1.1 Inutilisé
26E 1.Ø     poids fort idem (ØBB8Ø/BF4Ø en mode Text/Hires)
        1.1 Inutilisé
```

```
26F  1.Ø     nombre de lignes de l'écran
        1.1 Inutilisé

27Ø  1.Ø/1.1 Inutilisé
271  1.Ø/1.1 'couleur',dans b7,du curseur

272  1.Ø/1.1 Poids faible Timer 1 (gestion clavier)
273  1.Ø/1.1 Poids fort idem

274  1.Ø/1.1 Poids faible Timer 2 (clignotement curseur)
275  1.Ø/1.1 Poids fort idem

276  1.Ø/1.1 Poids faible timer 3 (utilsateur/WAIT)
277  1.Ø/1.1 Poids fort idem

278      1.1 Adresse de la 2 ème ligne d'écran
     1.Ø     Inutilisé
279      1.1 poids fort idem (#BBDØ/#BF9Ø pour Text/Hires)
     1.Ø     Inutilisé

27A      1.1 Adresse de la première ligne d'écran
     1.Ø     Inutilisé
27B      1.1 poids fort idem (#BBA8/#BF68 pour Text/Hires)
     1.Ø     Inutilisé

27C      1.1 Nombre de caractères à scroller
     1.Ø     Inutilisé
27D      1.1 poids fort idem (4Ø*(nombre de lignes-1))
     1.Ø     Inutilisé

27E      1.1 nombre de lignes à scroller
     1.Ø     Inutilisé

27F      1.1 nom du programme demandé
...
28F
27F  1.Ø     Inutilisé
28F

29Ø  1.Ø/1.1 Inutilisé
291  1.Ø/1.1 Inutilisé
292  1.Ø/1.1 Inutilisé
```

```
293     1.1 nom programme trouvé
...
2A3
293 1.Ø     Inutilisé
...
2A3

2A4 1.Ø/1.1 Inutilisé
2A5 1.Ø/1.1 Inutilisé
2A6 1.Ø/1.1 Inutilisé
2A7 1.Ø/1.1 Inutilisé

2A8     1.1 Entête,inutilisé
    1.Ø     Inutilisé
2A9     1.1 Entête,début poids faible
    1.Ø     Inutilisé
2AA     1.1 Entête,début poids fort
    1.Ø     Inutilisé
2AB     1.1 Entête,fin poids faible
    1.Ø     Inutilisé
2AC     1.1 Entête,fin poids fort
    1.Ø     Inutilisé
2AD     1.1 Entête,type (#ØØ basic,#8Ø bloc mémoire,#4Ø Tableau)
    1.Ø     Inutilisé
2AE     1.1 Entête,AUTO (si non nul)
    1.Ø     Inutilisé
2AF     1.1 Entête,drapeau chaine
    1.Ø     Inutilisé
2BØ     1.1 Entête,drapeau entier
    1.Ø     Inutilisé

2B1     1.1 Erreurs pendant lecture
    1.Ø     Inutilisé

2B2 1.Ø/1.1 Inutilisé
...
2BF

2CØ 1.Ø/1.1 bØ=1:Mode Hires.bØ=Ø:Mode Text.b1=1:mode RELEASE.b1=Ø:mode GRAB

2C1 1.Ø/1.1 Poids faible:Himem maxi,mode RELEASE (#98ØØ)
2C2 1.Ø/1.1 Poids fort idem

2C3 1.Ø/1.1 Mode d'adréssage du curseur Hires (1= relatif,Ø sinon)
```

```
2C4  1.Ø/1.1 Inutilisé
...
2DE

2DF  1.Ø/1.1 b7=Ø:pas de touche préssée.
             b7=1:bØ-b6 contient le code ASCII de la touche préssée

2EØ  1.Ø/1.1 Drapeau d'érreur,incrémenté si érreur pendant routines son/graph.

2E1  1.Ø/1.1 Param1,poids faible
2E2  1.Ø/1.1 Param1,poids fort
2E3  1.Ø/1.1 Param2,poids faible
2E4  1.Ø/1.1 Param2,poids fort
2E5  1.Ø/1.1 Param3,poids faible
2E6  1.Ø/1.1 Param3,poids fort
2E7  1.Ø/1.1 Param4,poids faible
2E8  1.Ø/1.1 Param4,poids fort

2E9  1.Ø/1.1 Inutilisé
...
2EF

2FØ  1.Ø/1.1 Comptage paramètres (syntaxe routines son/graphique)

2F1  1.Ø/1.1 b7=1:imprimante en service.b7=Ø:imprimante hors service

2F2  1.Ø/1.1 b7=Ø,LIST retourne à l'interpréteur.b7=1,retourne à l'appelant.

2F3  1.Ø/1.1 Inutilisé
2F4  1.Ø/1.1 b7=1,mode TRON.b7=Ø,mode TROFF.Sur 1.Ø,initialisé à chaque 'Ready'

2F5  1.Ø/1.1 Poids faible adresse !
2F6  1.Ø/1.1 Poids fort idem (normalement,$D2AØ/$D336)

2F7  1.Ø     'vidéo inverse'
         1.1 Inutilisé

2F8  1.Ø/1.1 Sauver coordonnée horizontale (PLOT...)
              1.Ø:Décalé de 1

2F9  1.Ø/1.1 Inutilisé
2FA  1.Ø/1.1 Inutilisé
2FB  1.Ø/1.1 $4C:JMP $xxxx,vecteur de la fonction &()
2FC  1.Ø/1.1 Poids faible adresse &()
```

2FD 1.Ø/1.1 Poids fort idem (normalement,#D2AØ/#D336)

2FE 1.Ø/1.1 Inutilisé
2FF 1.Ø/1.1 Inutilisé

D QUE RESTE-T-IL ?

1-Introduction

C'est ici un chapitre à l'aspect uniquement pratique: lorsqu'on programme en assembleur, on a constamment besoin de place pour mettre les variables systèmes, et évidemment pour implanter le programme lui même.

2-Où placer ses variables systèmes

-Dans la page zéro

Pour des raisons exposées précédemment, il est recommandé d'utiliser au maximum la page Ø pour ses variables de travail. On gagne ainsi environ 2Ø % de place dans les programmes et 1Ø % en rapidité.

On peut utiliser sans aucune précaution les zones suivantes: #ØØ-#ØC, #F3-#F9, ainsi que #16, #BB et #C1, soit 22 octets, qui couvrent l'immense majorité des besoins.

Si on a besoin de plus de place, on peut se servir de la semence du générateur aléatoire (#FA-#FE), à condition que #FA contiennent toujours #8Ø.

De même, le tampon clavier est utilisable dans sa totalité, mais attention au mode direct: il faut que TXTPTR retrouve quelque chose de cohérent au sortir de votre routine. La solution (brutale) existe, elle est utilisée d'ailleurs dans la V1.Ø pour le CLOAD: sortir en retournant directement à l'interpréteur (JMP #C9Ø3).

Si votre routine ne se sert pas de routines de la ROM, toute la page Ø est disponible sans restriction (sauf le pointeur #14-15, modifié par les IRQ's).

Si votre routine ne se sert pas de la ROM mais doit rendre la main au Basic, il faut faire attention aux quelques vecteurs ou variables qui ne doivent pas être touchés, c'est le cas de: #1Ø-#13, #1A-#1C, #21-#23, #3Ø-#32, #85-#87, #9A-#B1, #C2-#C5, #D7, #E2-#F2.

Si votre routine doit utiliser des sous programmes de la ROM, chaque cas est un cas particulier. Quelques conseils cependant: tous les accumulateurs flottants peuvent être modifiés par l'évaluation d'une expression, il ne faut donc pas s'en servir pour stocker ses propres données. Ils peuvent en revanche servir pour des valeurs temporaires, évidemment.

-Dans la page 2

Il y a beaucoup de place disponible dans la page 2, mais elle n'est pas très recommandée:

L'utilisation de la page 2 est très différente selon les versions de la ROM.

Dès l'instant que l'on ne travaille pas en page Ø, toutes les autres adresses sont équivalentes du point de vue temps ou longueur de programme. Il est donc préférable d'utiliser des variables intégrées au programme lui même. On réservera la page 2 aux données ou routines systèmes.

3-Où placer ses routines.

-Dans la page zéro

Placer ses routines dans la page zéro ne présente aucun intérêt, et est très difficile compte tenu du peu de place disponible.

-Dans la page un

Celà présente beaucoup de risque, et impose une gestion pour le moins rigoureuse de la pile...

Proscrit en tous cas si le programme doit rendre la main au BASIC.

En fait, cela n'a vraiment que des inconvénients ou presque, surtout avec 48 Ko de mémoire !!. Cette possibilité est signalée car elle a été vue dans un programme du commerce.

-Dans la page deux

La page 2 se prête bien à de courtes routines systèmes (détourner les IRQ, NMI etc...), car c'est une zone mémoire très stable, dans la mesure évidemment où l'on n'utilise que les octets libres.

-Dans la page quatre

La page quatre est très pratique, car c'est une zone très stable, puisqu'elle est entièrement libre et que la ROM n'y fait jamais appel.

Elle est donc recommandée pour y implanter ses propres routines.

Il y a hélas un gros défaut: tous les DOS existants s'en servent par nécessité comme amorce pour leurs propres besoins. La page 4 ne doit être utilisée sous aucn prétexte si la routine doit être utilisée un jour ou l'autre avec un lecteur de disquettes.

Ailleurs dans la mémoire

Il est heureusement possible d'implanter des routines n'importe où en

mémoire ou presque.

-La zone #0500-#97FF peut être utilisée sans problèmes de mode Text ou Hires, à condition de protéger le programme du Basic et de ses variables par un HIMEM adéquat.

-La zone #9800-#B4FF peut être utilisée sans aucune précaution en mode RELEASE, si le mode Hires n'est jamais appelé. En mode GRAB, il faudra protéger le programme par un HIMEM adéquat là aussi.

-Il existe des zones assez particulières: les jeux de caractères n'utilisent que 3 pages sur les 4 déplacées lors des passages Hires/Text.

Ces 2 zones sont protégées dans tous les modes.

Il s'agit des zones #B400-#B4FF et #B800-#B8FF, qui sont décalées mais non altérées en #9800-#98FF et #9C00-#9CFF lors du passage en mode haute résolution, puis restituées lors d'un nouveau passage en mode Text.

Ces zones sont très pratiques pour y placer des routines relogeables, ou des routines à n'utiliser que dans un seul mode.

On peut utiliser éventuellement le deuxième jeu de caractères (#B900-#BB7F) pour stocker des programmes, il est lui aussi déplacé. Mais attention aux NMI qui régénèrent le jeu de caractère (zone #B900-#BAFF).

-Enfin, la zone #BFE0-#BFFF est inutilisée, quel que soit le mode. Malheureusement, une bogue de l'affichage en double hauteur risque de détruire cette zone, et rend son utilisation périlleuse.

CHAPITRE VI

LA ROM

A) INTRODUCTION

1-Introduction

La partie la plus longue de ce livre est consacrée au listing exhaustif et surtout commenté, de la ROM.

Afin de faciliter la recherche d'une routine donnée, la linéarité de la ROM n'a jamais été rompue. Ainsi on trouvera dans l'ordre toutes les routines de #C000 à #FFFF.

On peut assez facilement - et artificiellement - séparer la ROM en diverses zones, telles l'interpréteur, les routines de calcul etc... En ceci, la construction même de la ROM nous a aidé.

Il est vrai qu'on n'écrit pas une ROM sans une certaine rigueur. L'organisation de la ROM est une séquelle des premiers micros-ordinateurs: le haut de la mémoire comportait le moniteur, ensemble de routines particulières au système lui-même, le reste de l'espace mémoire étant réservé au langage. Cette disposition est maintenant réservée aux plus gros ordinateurs.

Il est de même logique que le Basic commence par l'interpréteur, suivi par les commandes vitales, puis par les routines de calculs. Cet ensemble d'environ 8Ko forme le noyau Microsoft, auxquels ont été rajoutées les routines graphiques et sonores. Cette architecture se retrouve pour tout ordinateur avec un Basic d'origine Microsoft. Simplement, avec un Z80 par exemple, les routines systèmes sont au début, suivies par le langage, car le vecteur de RESET est en #00 et non en #FFFF: toujours la philosophie du moniteur.

B) LES TABLES

1-Tables d'adresse

VECTEURS SYSTEMES

Ces vecteurs ne sont jamais utilisés, mais sont pratiques pour l'utilisateur.

```
C000 JMP $EA59   C000 JMP $ECCC  ;Initialisation à froid BASIC
C003 JMP $C475   C003 JMP $C471  ;Initialisation à chaud BASIC
```

TABLE D'ADRESSE

Les adresses des commandes sont stockées avec leur valeur -1 car elles sont appelées par un RTS.

```
C006 C006 BYT #C941-1/#C973-1   ;END
C008 C008 BYT #C6A5-1/#C692-1   ;EDIT
C00A C00A BYT #CFE4-1/#E987-1   ;INVERSE/STORE
C00C C00C BYT #CFE4-1/#E9D1-1   ;NORMAL/RECALL
C00E C00E BYT #CC8C-1/#CD16-1   ;TRON
C010 C010 BYT #CC8F-1/#CD19-1   ;TROFF
C012 C012 BYT #C9E0-1/#CA12-1   ;POP
C014 C014 BYT #D9C6-1/#DA51-1   ;PLOT
C016 C016 BYT #DA16-1/#DAA1-1   ;PULL
C018 C018 BYT #D937-1/#D9DE-1   ;LORES
C01A C01A BYT #D8AC-1/#D967-1   ;DOKE
C01C C01C BYT #D9FA-1/#DA85-1   ;REPEAT
C01E C01E BYT #DA16-1/#DAA1-1   ;UNTIL
C020 C020 BYT #C841-1/#C855-1   ;FOR
C022 C022 BYT #C824-1/#C7FD-1   ;LLIST
C024 C024 BYT #C832-1/#C809-1   ;LPRINT
C026 C026 BYT #CE0C-1/#CE98-1   ;NEXT
C028 C028 BYT #CA0A-1/#CA3C-1   ;DATA
C02A C02A BYT #CCC9-1/#CD55-1   ;INPUT
C02C C02C BYT #D0F2-1/#D17E-1   ;DIM
C02E C02E BYT #CC0A-1/#CCCE-1   ;CLS
C030 C030 BYT #CCFD-1/#CD89-1   ;READ
C032 C032 BYT #CAD2-1/#CB1C-1   ;LET
C034 C034 BYT #C9B3-1/#C9E5-1   ;GOTO
C036 C036 BYT #C98B-1/#C9BD-1   ;RUN
C038 C038 BYT #CA3E-1/#CA70-1   ;IF
C03A C03A BYT #C91F-1/#C952-1   ;RESTORE
C03C C03C BYT #C996-1/#C9C8-1   ;GOSUB
C03E C03E BYT #C9E0-1/#CA12-1   ;RETURN
C040 C040 BYT #CA61-1/#CA99-1   ;REM
C042 C042 BYT #E95B-1/#EBCE-1   ;HIMEM
C044 C044 BYT #E974-1/#EBE7-1   ;GRAB
C046 C046 BYT #E994-1/#EC0C-1   ;RELEASE
C048 C048 BYT #E9A9-1/#EC21-1   ;TEXT
C04A C04A BYT #E9BB-1/#EC33-1   ;HIRES
```

```
C04C C04C BYT #F415-1/#FAB5-1   ;SHOOT
C04E C04E BYT #F418-1/#FACB-1   ;EXPLODE
C050 C050 BYT #F41B-1/#FAE1-1   ;ZAP
C052 C052 BYT #F412-1/#FA9F-1   ;PING
C054 C054 BYT #E889-1/#EAFC-1   ;SOUND
C056 C056 BYT #E889-1/#EAFC-1   ;MUSIC
C058 C058 BYT #E889-1/#EAFC-1   ;PLAY
C05A C05A BYT #E87D-1/#EAF0-1   ;CURSET
C05C C05C BYT #E87D-1/#EAF0-1   ;CURMOV
C05E C05E BYT #E87D-1/#EAF0-1   ;DRAW
C060 C060 BYT #E87D-1/#EAF0-1   ;CIRCLE
C062 C062 BYT #E87D-1/#EAF0-1   ;PATTERN
C064 C064 BYT #E87D-1/#EAF0-1   ;FILL
C066 C066 BYT #E87D-1/#EAF0-1.  ;CHAR
C068 C068 BYT #E889-1/#EAFC-1   ;PAPER
C06A C06A BYT #E889-1/#EAFC-1   ;INK
C06C C06C BYT #C93F-1/#C971-1   ;STOP
C06E C06E BYT #CA78-1/#CAC2-1   ;ON
C070 C070 BYT #D89D-1/#D958-1   ;WAIT
C072 C072 BYT #E7AA-1/#E85B-1   ;CLOAD
C074 C074 BYT #E7DB-1/#E909-1   ;CSAVE
C076 C076 BYT #D401-1/#D4BA-1   ;DEF
C078 C078 BYT #D894-1/#D94F-1   ;POKE
C07A C07A BYT #CB61-1/#CBAB-1   ;PRINT
C07C C07C BYT #C96E-1/#C9A0-1   ;CONT
C07E C07E BYT #C773-1/#C748-1   ;LIST
C080 C080 BYT #C738-1/#C70D-1   ;CLEAR
C082 C082 BYT #CCBA-1/#CD46-1   ;GET
C084 C084 BYT #E80D-1/#E946-1   ;CALL
C086 C086 BYT #CC89-1/#CD13-1   ;!
C088 C088 BYT #C719-1/#C6EE-1   ;NEW
```

TABLE D'ADRESSE DES FONCTIONS

```
C08A C08A BYT #DF12/#DF21   ;SGN
C08C C08C BYT #DFA5/#DFBD   ;INT
C08E C08E BYT #DF31/#DF49   ;ABS
C090 C090 BYT #0021/#0021   ;USR
C092 C092 BYT #D3D6/#D47E   ;FRE
C094 C094 BYT #D3FA/#D4A6   ;POS
C096 C096 BYT #D917/#D9B5   ;HEX$
C098 C098 BYT #02FB/#02FB   ;&
C09A C09A BYT #E22A/#E22E   ;SQR
```

```
C09C C09C BYT #E34B/#E34F   ;RND
C09E C09E BYT #DC79/#DCAF   ;LN
C0A0 C0A0 BYT #E2A6/#E2AA   ;EXP
C0A2 C0A2 BYT #E387/#E38B   ;COS
C0A4 C0A4 BYT #E38E/#E392   ;SIN
C0A6 C0A6 BYT #E3D7/#E3DB   ;TAN
C0A8 C0A8 BYT #E43B/#E43F   ;ATN
C0AA C0AA BYT #D87D/#D938   ;PEEK
C0AC C0AC BYT #D8C8/#D983   ;DEEK
C0AE C0AE BYT #DDD0/#DDD4   ;LOG
C0B0 C0B0 BYT #D7EB/#D8A6   ;LEN
C0B2 C0B2 BYT #D4D8/#D593   ;STR$
C0B4 C0B4 BYT #D81C/#D8D7   ;VAL
C0B6 C0B6 BYT #D7FA/#D8B5   ;ASC
C0B8 C0B8 BYT #D75B/#D816   ;CHR$
C0BA C0BA BYT #D8EE/#DE77   ;PI
C0BC C0BC BYT #DF00/#DF0F   ;TRUE
C0BE C0BE BYT #DEFC/#DF0B   ;FALSE
C0C0 C0C0 BYT #DA4F/#DADA   ;KEY$
C0C2 C0C2 BYT #D9B4/#DA3F   ;SCRN
C0C4 C0C4 BYT #E9CD/#EC45   ;POINT
C0C6 C0C6 BYT #D76F/#D82A   ;LEFT$
C0C8 C0C8 BYT #D79B/#D856   ;RIGHT$
C0CA C0CA BYT #D7A6/#D861   ;MID$
```

TABLE D'ADRESSE DES OPERATEURS

Les adresses sont stockées -1 car les opérateurs sont appelés par RTS. Rappel:A et Z doivent être placés selon l'exposant de l'opérande dans ACC1 pour les opérateurs de calcul.

Voici l'ordre de priorité de l'exécution (du plus prioritaire au moins prioritaire):

^, changement de signe, * ou /, + ou -, >=<, NOT, AND, OR.

Les codes de priorité auraient aussi bien pu être échelonnés de 1 à 10 par exemple, sans perturber le système. Ces codes sont donc bien mystérieux...

```
C0CC C0CC 00 BYT #79
C0CD C0CD    BYT #DA9A-1/#DB25-1  ;+
C0CF C0CF 03 BYT #79
C0D0 C0D0    BYT #DA83-1/#DB0E-1  ;-
C0D2 C0D2 06 BYT #7B
C0D3 C0D3    BYT #DCBA-1/#DCF0-1  ;*
```

```
C0D5 C0D5 09 BYT #7B
C0D6 C0D6    BYT #DDE3-1/#DDE7-1  ;/
C0D8 C0D8 0C BYT #7F
C0D9 C0D9    BYT #E234-1/#E238-1  ;^
C0DB C0DB 0F BYT #50
C0DC C0DC    BYT #D05A-1/#D0E6-1  ;AND
C0DE C0DE 12 BYT #46
C0DF C0DF    BYT #D057-1/#D0E3-1  ;OR
C0E1 C0E1 15 BYT #7D
C0E2 C0E2    BYT #E26D-1/#E271-1  ;Changement de signe
C0E4 C0E4 18 BYT #5A
C0E5 C0E5    BYT #CFB0-1/#D03C-1  ;NOT
C0E7 C0E7 1B BYT #64
C0E8 C0E8    BYT #D087-1/#D113-1   )=(
```

2-Table des mots-clés

TABLE DES COMMANDES

Sont donnés dans l'ordre: adresse V1.0, adresse V1.1, code du mot clé ou token en hexadécimal, et enfin le mot-clé lui-même.

```
C0EA C0EA 80 BYT 'EN','D'+#80
C0ED C0ED 81 BYT 'EDI','T'+#80
C0F1 .... 82 BYT 'INVERS','E'+#80
.... C0F1 82 BYT 'STOR','E'+#80
C0F8 .... 83 BYT 'NORMA','L'+#80
.... C0F6 83 BYT 'RECAL','L'+#80
C0FE C0FC 84 BYT 'TRO','N'+#80
C102 C100 85 BYT 'TROF','F'+#80
C107 C105 86 BYT 'PO','P'+#80
C10A C108 87 BYT 'PLO','T'+#80
C10E C10C 88 BYT 'PUL','L'+#80
C112 C110 89 BYT 'LORE','S'+#80
C117 C115 8A BYT 'DOK','E'+#80
C11B C119 8B BYT 'REPEA','T'+#80
C121 C11F 8C BYT 'UNTI','L'+#80
C126 C124 8D BYT 'FO','R'+#80
C129 C127 8E BYT 'LLIS','T'+#80
C12E C12C 8F BYT 'LPRIN','T'+#80
C134 C132 90 BYT 'NEX','T'+#80
C138 C136 91 BYT 'DAT','A'+#80
```

```
C13C C13A 92 BYT 'INPU','T'+$80
C141 C13F 93 BYT 'DI','M'+$80
C144 C142 94 BYT 'CL','S'+$80
C147 C145 95 BYT 'REA','D'+$80
C14B C149 96 BYT 'LE','T'+$80
C14E C14C 97 BYT 'GOT','O'+$80
C152 C150 98 BYT 'RU','N'+$80
C155 C153 99 BYT 'I','F'+$80
C157 C155 9A BYT 'RESTOR','E'+$80
C15E C15C 9B BYT 'GOSU','B'+$80
C163 C161 9C BYT 'RETUR','N'+$80
C169 C167 9D BYT 'RE','M'+$80
C16C C16A 9E BYT 'HIME','M'+$80
C171 C16F 9F BYT 'GRA','B'+$80
C175 C173 A0 BYT 'RELEAS','E'+$80
C17C C17A A1 BYT 'TEX','T'+$80
C180 C17E A2 BYT 'HIRE','S'+$80
C185 C183 A3 BYT 'SHOO','T'+$80
C18A C188 A4 BYT 'EXPLOD','E'+$80
C191 C18F A5 BYT 'ZA','P'+$80
C194 C192 A6 BYT 'PIN','G'+$80
C198 C196 A7 BYT 'SOUN','D'+$80
C19D C19B A8 BYT 'MUSI','C'+$80
C1A2 C1A0 A9 BYT 'PLA','Y'+$80
C1A6 C1A4 AA BYT 'CURSE','T'+$80
C1AC C1AA AB BYT 'CURMO','V'+$80
C1B2 C1B0 AC BYT 'DRA','W'+$80
C1B6 C1B4 AD BYT 'CIRCL','E'+$80
C1BC C1BA AE BYT 'PATTER','N'+$80
C1C3 C1C1 AF BYT 'FIL','L'+$80
C1C7 C1C5 B0 BYT 'CHA','R'+$80
C1CB C1C9 B1 BYT 'PAPE','R'+$80
C1D0 C1CE B2 BYT 'IN','K'+$80
C1D3 C1D1 B3 BYT 'STO','P'+$80
C1D7 C1D5 B4 BYT 'O','N'+$80
C1D9 C1D7 B5 BYT 'WAI','T'+$80
C1DD C1DB B6 BYT 'CLOA','D'+$80
C1E2 C1E0 B7 BYT 'CSAV','E'+$80
C1E7 C1E5 B8 BYT 'DE','F'+$80
C1EA C1E8 B9 BYT 'POK','E'+$80
C1EE C1EC BA BYT 'PRIN','T'+$80
C1F3 C1F1 BB BYT 'CON','T'+$80
C1F7 C1F5 BC BYT 'LIS','T'+$80
C1FB C1F9 BD BYT 'CLEA','R'+$80
```

```
C200 C1FE BE BYT 'GE','T'+$80
C203 C201 BF BYT 'CAL','L'+$80
C207 C205 C0 BYT '!'+$80
C208 C206 C1 BYT 'NE','W'+$80
```

TABLE MOTS CLES DIVERS

```
C20B C209 C2 BYT 'TAB','('+$80
C20F C20D C3 BYT 'T','O'+$80
C211 C20F C4 BYT 'F','N'+$80
C213 C211 C5 BYT 'SPC','('+$80
C217 C215 C6 BYT '@'+$80
C218 C216 C7 BYT 'AUT','O'+$80
C21C C21A C8 BYT 'ELS','E'+$80
C220 C21E C9 BYT 'THE','N'+$80
C224 C222 CA BYT 'NO','T'+$80
C227 C225 CB BYT 'STE','P'+$80
```

TABLE DES OPERATEURS

```
C22B C229 CC BYT '+'+$80
C22C C22A CD BYT '-'+$80
C22D C22B CE BYT '*'+$80
C22E C22C CF BYT '/'+$80
C22F C22D D0 BYT '^'+$80
C230 C22E D1 BYT 'AN','D'+$80
C233 C231 D2 BYT 'O','R'+$80
C235 C233 D3 BYT ')'+$80
C236 C234 D4 BYT '='+$80
C237 C235 D5 BYT '<'+$80
```

TABLES DES FONCTIONS

Reste: le mot clé GO traine encore dans la V1.0. Certains Basic acceptent le mot GO TO comme GOTO, ce qui a failli être le cas de l'oric. Il est vrai que c'est bien peu utile.

```
C238 C236 D6 BYT 'SG','N'+$80
C23B C239 D7 BYT 'IN','T'+$80
C23E C23C D8 BYT 'AB','S'+$80
C241 C23F D9 BYT 'US','R'+$80
```

```
C244 C242 DA BYT 'FR','E'+$80
C247 C245 DB BYT 'PO','S'+$80
C24A C248 DC BYT 'HEX','$'+$80
C24E C24C DD BYT 'L'+$80
C24F C24D DE BYT 'SP','R'+$80
C252 C250 DF BYT 'RN','D'+$80
C255 C253 E0 BYT 'L','N'+$80
C257 C255 E1 BYT 'EX','P'+$80
C25A C258 E2 BYT 'CO','S'+$80
C25D C25B E3 BYT 'SI','N'+$80
C260 C25E E4 BYT 'TA','N'+$80
C263 C261 E5 BYT 'AT','N'+$80
C266 C264 E6 BYT 'PEE','K'+$80
C26A C268 E7 BYT 'DEE','K'+$80
C26E C26C E8 BYT 'LO','G'+$80
C271 C26F E9 BYT 'LE','N'+$80
C274 C272 EA BYT 'STR','$'+$80
C278 C276 EB BYT 'VA','L'+$80
C27B C279 EC BYT 'AS','C'+$80
C27E C27C ED BYT 'CHR','$'+$80
C282 C280 EE BYT 'P','I'+$80
C284 C282 EF BYT 'TRU','E'+$80
C288 C286 F0 BYT 'FALS','E'+$80
C28D C28B F1 BYT 'KEY','$'+$80
C291 C28F F2 BYT 'SCR','N'+$80
C295 C293 F3 BYT 'POIN','T'+$80
C29A C298 F4 BYT 'LEFT','$'+$80
C29F C29D F5 BYT 'RIGHT','$'+$80
C2A5 C2A3 F6 BYT 'MID','$'+$80
C2A9 .... F7 BYT 'G','O'+$80
C2AB C2A7    BYT $00
```

3-Les messages d'érreur

TABLE DES MESSAGES D'ERREUR

Sont donnés dans l'ordre: adresse du méssage pour V1.0, adresse du message pour V1.1, déplacement dans la table (utilisé pour déclencher les messages), adresse de déclenchement V1.0, adresse de déclenchement V1.1, et enfin le message lui-même.
Astuce: 'DISP TYPE MISMATCH' et 'TYPE MISMATCH' sont regroupés.

```
C2AC C2A8 +ØØ  CE1E/CEAA BYT 'NEXT WITHOUT FO','R'+#8Ø
C2BC C2B8 +1Ø  CFE4/DØ7Ø BYT 'SYNTA','X'+#8Ø
C2C2 C2BE +16  C9EE/CA2Ø BYT 'RETURN WITHOUT GOSU','B'+#8Ø
C2D6 C2D2 +2A  ..../D35C BYT 'OUT OF DAT','A'+#8Ø
C2E1 C2DD +35  D2AØ/D336 BYT 'ILLEGAL QUANTIT','Y'+#8Ø
C2F1 C2ED +45  DBEØ/DC39 BYT 'OVERFLO','W'+#8Ø
C2F9 C2F5 +4D  C483/C47C BYT 'OUT OF MEMOR','Y'+#8Ø
C3Ø6 C3Ø2 +5A  C9F1/CA23 BYT 'UNDEF'D STATEMEN','T'+#8Ø
C317 C313 +6B  D29D/D333 BYT 'BAD SUBSCRIP','T'+#8Ø
C324 C32Ø +78  D2A5/D33B BYT 'REDIM'D ARRA','Y'+#8Ø
C331 C32D +85  DE5B/DE5F BYT 'DIVISION BY ZER','O'+#8Ø
C341 C33D +95  D41E/D4D7 BYT 'ILLEGAL DIREC','T'+#8Ø
C34F C34B +A3  DA72/DAFD BYT 'DISP '
C354 C35Ø +A8  CE86/CF12 BYT 'TYPE MISMATC','H'+#8Ø
C361 C35D +B5  D6C7/D782 BYT 'STRING TOO LON','G'+#8Ø
C37Ø C36C +C4  D53F/D5FA BYT 'FORMULA TOO COMPLE','X'+#8Ø
C383 C37F +D7  ..../.... BYT 'CAN'T CONTINU','E'+#8Ø
C391 C38D +E5  D421/D4DA BYT 'UNDEF'D FUNCTIO','N'+#8Ø
C3A1 C39D +F5  DA22/DAAD BYT 'BAD UNTI','L'+#8Ø
```

DIVERS MESSAGES

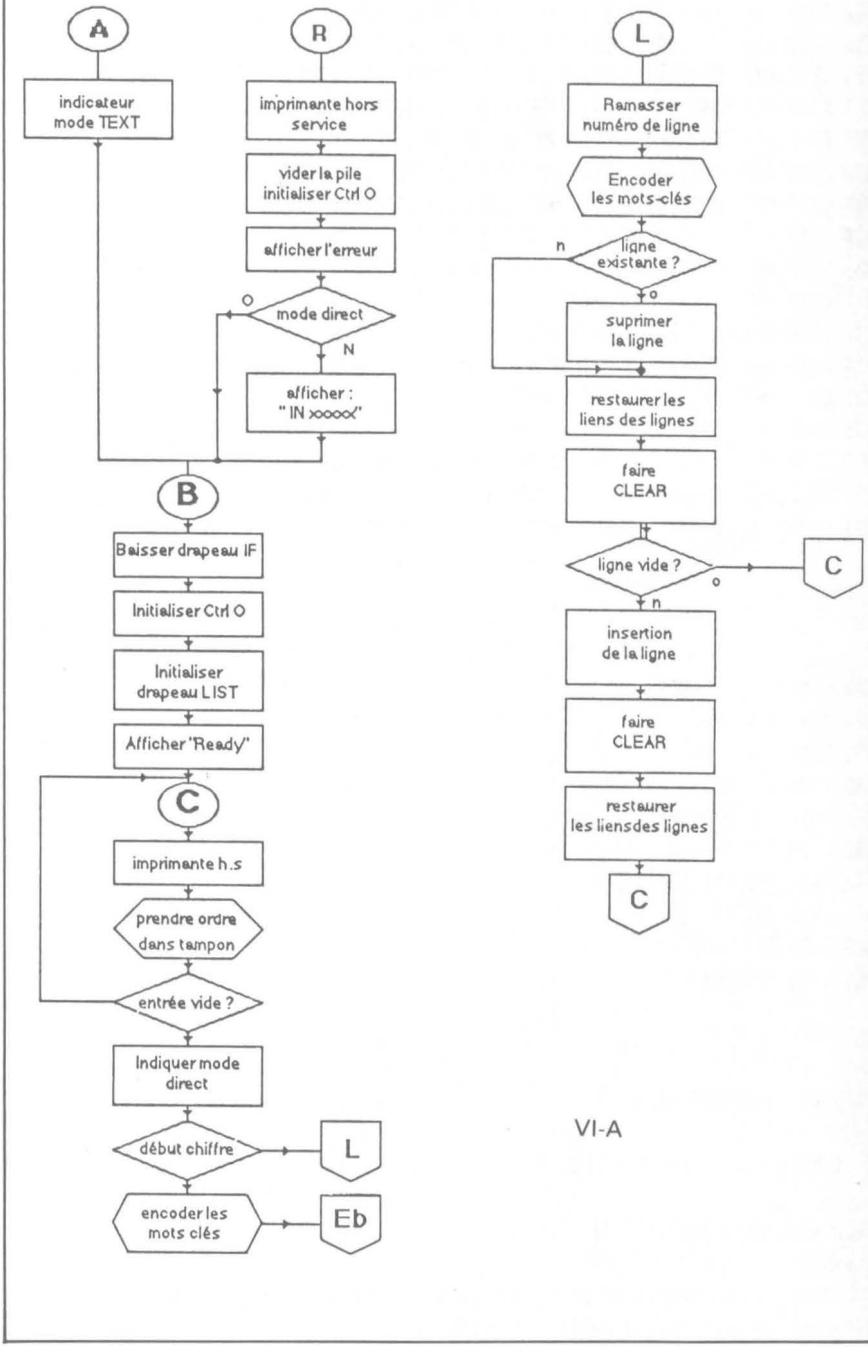
```
C3AA C3A6 BYT ' ERROR'
C3BØ C3AC BYT #ØØ
C3B1 C3AD BYT ' IN '
C3B5 C3B1 BYT #ØØ
C3B6 C3B2 BYT #ØD,#ØA
C3B8 C3B4 BYT 'Ready '
C3BE C3BA BYT #ØD,#ØA,#ØØ
C3C1 C3BD BYT #ØD,#ØA
C3C3 C3BF BYT ' BREAK'
C3C9 C3C5 BYT #ØØ
```

C) L'INTERPRETEUR BASIC

1-STRUCTURE DE L'INTERPRETEUR

La strucutre genérale de l'interpréteur est décrite par les organigrammes suivants.

Les adresses données le sont pour la V1.1, ainsi que les organigrammes qui diffèrent un peu pour la gestion du ELSE ou de "'".


A
indicateur mode TEXT
R
imprimante hors service
vider la pile initialiser Ctrl O
afficher l'erreur
mode direct
O
N
afficher : " IN xxxxx"
B
Baisser drapeau IF
Initialiser Ctrl O
Initialiser drapeau LIST
Afficher 'Ready"
C
imprimante h.s
prendre ordre dans tampon
entrée vide ?
Indiquer mode direct
début chiffre
L
encoder les mots clés
Eb
L
Ramasser numéro de ligne
Encoder les mots-clés
ligne existante ?
n
o
suprimer la ligne
restaurer les liens des lignes
faire CLEAR
ligne vide ?
o
C
n
insertion de la ligne
faire CLEAR
restaurer les liensdes lignes
C

VI-A

```
C2AC C2A8 +ØØ  CE1E/CEAA BYT 'NEXT WITHOUT FO','R'+#8Ø
C2BC C2B8 +1Ø  CFE4/DØ7Ø BYT 'SYNTA','X'+#8Ø
C2C2 C2BE +16  C9EE/CA2Ø BYT 'RETURN WITHOUT GOSU','B'+#8Ø
C2D6 C2D2 +2A  ..../D35C BYT 'OUT OF DAT','A'+#8Ø
C2E1 C2DD +35  D2AØ/D336 BYT 'ILLEGAL QUANTIT','Y'+#8Ø
C2F1 C2ED +45  DBEØ/DC39 BYT 'OVERFLO','W'+#8Ø
C2F9 C2F5 +4D  C483/C47C BYT 'OUT OF MEMOR','Y'+#8Ø
C3Ø6 C3Ø2 +5A  C9F1/CA23 BYT 'UNDEF'D STATEMEN','T'+#8Ø
C317 C313 +6B  D29D/D333 BYT 'BAD SUBSCRIP','T'+#8Ø
C324 C32Ø +78  D2A5/D33B BYT 'REDIM'D ARRA','Y'+#8Ø
C331 C32D +85  DE5B/DE5F BYT 'DIVISION BY ZER','O'+#8Ø
C341 C33D +95  D41E/D4D7 BYT 'ILLEGAL DIREC','T'+#8Ø
C34F C34B +A3  DA72/DAFD BYT 'DISP '
C354 C35Ø +A8  CE86/CF12 BYT 'TYPE MISMATC','H'+#8Ø
C361 C35D +B5  D6C7/D782 BYT 'STRING TOO LON','G'+#8Ø
C37Ø C36C +C4  D53F/D5FA BYT 'FORMULA TOO COMPLE','X'+#8Ø
C383 C37F +D7  ..../.... BYT 'CAN'T CONTINU','E'+#8Ø
C391 C38D +E5  D421/D4DA BYT 'UNDEF'D FUNCTIO','N'+#8Ø
C3A1 C39D +F5  DA22/DAAD BYT 'BAD UNTI','L'+#8Ø
```

DIVERS MESSAGES

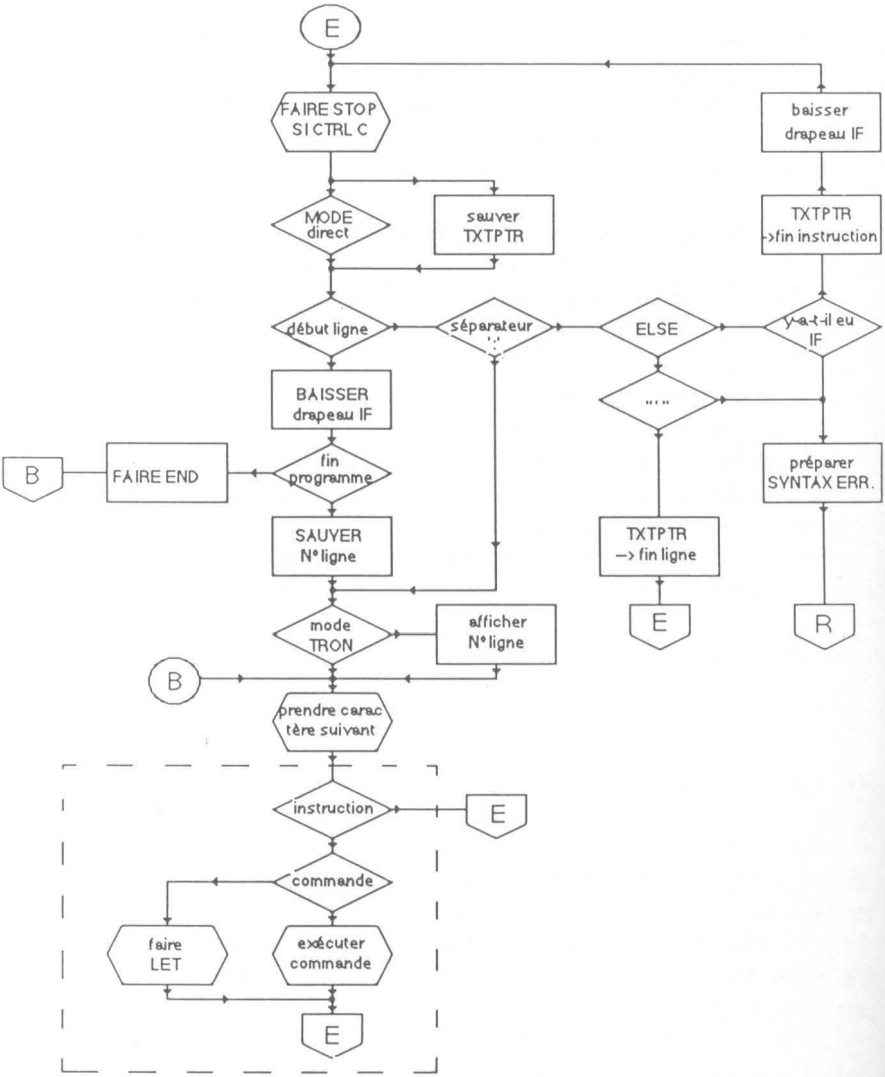
```
C3AA C3A6 BYT ' ERROR'
C3BØ C3AC BYT #ØØ
C3B1 C3AD BYT ' IN '
C3B5 C3B1 BYT #ØØ
C3B6 C3B2 BYT #ØD,#ØA
C3B8 C3B4 BYT 'Ready '
C3BE C3BA BYT #ØD,#ØA,#ØØ
C3C1 C3BD BYT #ØD,#ØA
C3C3 C3BF BYT ' BREAK'
C3C9 C3C5 BYT #ØØ
```

C) L'INTERPRETEUR BASIC

1-STRUCTURE DE L'INTERPRETEUR

La strucutre genérale de l'interpréteur est décrite par les organigrammes suivants.

Les adresses données le sont pour la V1.1, ainsi que les organigrammes qui diffèrent un peu pour la gestion du ELSE ou de "'".


E
FAIRE STOP SI CTRL C
MODE direct
sauver TXTPTR
début ligne
séparateur ':'
ELSE
y-a-t-il eu IF
baisser drapeau IF
TXTPTR ->fin instruction
BAISSER drapeau IF
fin programme
FAIRE END
B
SAUVER N°ligne
mode TRON
afficher N°ligne
B
prendre caractère suivant
instruction
E
commande
faire LET
exécuter commande
E
TXTPTR -> fin ligne
E
préparer SYNTAX ERR.
R

VI-B

2-La routine sort par JMP #ØØE2 qui saute la commande, et positionne A, Z et C selon le premier caractère du paramètre. Le RTS de #ØØE2 renvoie à l'adresse de la commande, et le RTS de la commande est en fait celui de #C8FE/C915.(Y vaut aussi le double du token, ce qui est pratique pour distinguer des commandes ayant la même adresse, Cf POP).

Le fait que A, Z et C soient positionnés correctement est très pratique pour détecter la présence de paramètre. Par exemple la commande NEW commence par un BNE #C718/#C6EE, qui signifie de sortir directement si un paramètre est présent, ce qui est incorrect.

Le message d'erreur sera déclenché après le retour en E: prenons l'exemple de la commande NEW A. L'interpréteur détecte et exécute le NEW, et retourne directement en E. Comme le caractère pointé par TXTPTR n'est pas un Ø, ce n'est pas le début de ligne, et on veut donc un ':', ou une REM abrégée, éventuellement un ELSE. Si ce n'est pas le cas, on déclenche alors l'erreur.

Ce procédé est très pratique: lorsqu'il y a trop de paramètres, il est inutile de générer directement un message d'erreur, il suffit de sortir prématurément de la commande. Notons que ! est une commande normale, et donc que A, Z et C sont correctement positionnés.

L'analyse de syntaxe doit être rigoureuse, et faite de telle sorte qu'une commande fausse syntaxiquement ne s'exécute pas. Par exemple, si on supprimait le BNE au début du NEW, NEW A déclencherait bien SYNTAX ERROR, mais la commande serait exécutée d'abord, ce qui est génant pour des commandes 'dangereuses'. Ceci l'est moins pour des commandes peu 'dangereuses' comme RESTORE, qui ne comporte donc aucune analyse de syntaxe 'préventive'.

Pour rompre l'exécution 'linéaire' d'un programme, il suffit de modifier dans la commande TXTPTR. C'est ainsi qu'agissent GOTO, GOSUB, NEXT, UNTIL, THEN, ELSE etc.

2-Listing

RECHERCHE DE BLOCS BASICS SUR LA PILE

Entrée: #B9=Ø si pas de variable précise à chercher.Sinon #B8-9 contient l'adresse de la variable de boucle.

Sortie: Z=Ø si pas bloc FOR correspondant trouvé,A contient alors le code du bloc trouvé (ØØ si le sommet de la pile est atteint).

Z=1 si un bloc FOR a été trouvé,correspondant à l'adresse de la variable précisée.

X contient la position du pointeur de pile,ajustée sur le code du bloc trouvé.

Y inchangé.

```
C3CA  TSX          C3C6  TSX          Prendre le pointeur de pile
C3CB  INX          C3C7  INX
C3CC  INX          C3C8  INX          et sauter l'adresse de retour
C3CD  INX          C3C9  INX
C3CE  INX          C3CA  INX          et l'adresse de retour à l'interpréteur
C3CF  LDA #101,X   C3CB  LDA #101,X   prendre le code du bloc BASIC
C3D2  CMP #8D      C3CE  CMP #8D      est-ce FOR ?
C3D4  BNE C3F7     C3D0  BNE C3F3     non,sortir Z=0
C3D6  LDA B9       C3D2  LDA B9       prendre poids fort adresse de la variable
C3D8  BNE C3E4     C3D4  BNE C3E0     si non nul,aller comparer
C3DA  LDA #102,X   C3D6  LDA #102,X   sinon,prendre ce bloc et donc
C3DD  STA B8       C3D9  STA B8       prendre l'adresse de la variable
C3DF  LDA #103,X   C3DB  LDA #103,X   de boucle
C3E2  STA B9       C3DE  STA B9       puis faire une comparaison bidon.
C3E4  CMP #103,X   C3E0  CMP #103,X   comparer poids fort variable à la pile
C3E7  BNE C3F0     C3E3  BNE C3EC     et chercher un autre bloc si pas bon
C3E9  LDA B8       C3E5  LDA B8       idem pour le poids faible
C3EB  CMP #102,X   C3E7  CMP #102,X
C3EE  BEQ C3F7     C3EA  BEQ C3F3     si ok,sortir avec Z=1
C3F0  TXA          C3EC  TXA          passer au bloc suivant:
C3F1  CLC          C3ED  CLC          ajouter la longueur d'un bloc
C3F2  ADC #12      C3EE  ADC #12      au pointeur de pile
C3F4  TAX          C3F0  TAX
C3F5  BNE C3CF     C3F1  BNE C3CB     et recommencer...
C3F7  RTS          C3F3  RTS
```

DECALER UN BLOC MEMOIRE VERS LE HAUT

Entrée: #CE-#CF=premier octet du bloc à décaler,
#C9-#CA=dernier octet du bloc.
#C7-#C8 et AY:adresse cible du bloc (soit le haut du bloc)

Sortie: OUT OF MEMORY si l'adresse cible est plus haute que les chaines (#A2)
Sinon,#C7-#C8 point sur le nouveau début du bloc -#100.
Le haut des tableaux (#A0) est placé à la nouvelle fin du Bloc.

Principe: puisqu'on décale vers le haut,il faut partir du haut du bloc vers le bas sous peine d'écrire sur le bloc qu'on est en train de décaler.
Les pointeurs sont ajustés de manière à transférer d'abord une fraction de page,et ensuite un nombre entier de pages.

Utilisation: si on veut se servir de la routine pour faire un décalage quelconque,il est préférable de l'attaquer en #C3FF/#C3FB,pour éviter le OUT OF

MEMORY et l'ajustement du pointeur de tableau.

```
C3F8  JSR $C448   C3F4  JSR $C444   vérifier place disponible
C3FB  STA A0      C3F7  STA A0      et sauver nouveau
C3FD  STY A1      C3F9  STY A1      top des tableaux
C3FF  SEC         C3FB  SEC         calcul de la taille du bloc à décaler.
C400  LDA C9      C3FC  LDA C9
C402  SBC CE      C3FE  SBC CE
C404  STA 91      C400  STA 91      Poids faible de la longueur en #91
C406  TAY         C402  TAY         et dans Y
C407  LDA CA      C403  LDA CA
C409  SBC CF      C405  SBC CF      Poids fort de la longueur en X
C40B  TAX         C407  TAX         YX contient la taille du bloc
C40C  INX         C408  INX         ajuster le nombre total de pages
C40D  TYA         C409  TYA         si il faut décaler un nombre entier
C40E  BEQ C433    C40A  BEQ C42F    de pages, on saute le décalage de Y octets
C410  LDA C9      C40C  LDA C9
C412  SEC         C40E  SEC         ajuster #C9-#CA pour obtenir
C413  SBC 91      C40F  SBC 91      un nombre entier de blocs
C415  STA C9      C411  STA C9
C417  BCS C41C    C413  BCS C418
C419  DEC CA      C415  DEC CA
C41B  SEC         C417  SEC         et agir de même pour
C41C  LDA C7      C418  LDA C7      le pointeur cible
C41E  SBC 91      C41A  SBC 91
C420  STA C7      C41C  STA C7
C422  BCS C42C    C41E  BCS C428
C424  DEC C8      C420  DEC C8
C426  BCC C42C    C422  BCC C428    inconditionnel

C428  LDA (C9),Y  C424  LDA (C9),Y  Décaler Y octets
C42A  STA (C7),Y  C426  STA (C7),Y  vers le bloc cible
C42C  DEY         C428  DEY
C42D  BNE C428    C429  BNE C424
C42F  LDA (C9),Y  C42B  LDA (C9),Y
C431  STA (C7),Y  C42D  STA (C7),Y  Transfert du dernier octet (pour Y=0)
C433  DEC CA      C42F  DEC CA      page suivante bloc source
C435  DEC C8      C431  DEC C8      et bloc cible
C437  DEX         C433  DEX         et décompter le nombre de pages.
C438  BNE C42C    C434  BNE C428    puis recommencer encore un bloc si pas fini
C43A  RTS         C436  RTS
```

VERIFIER PLACE SUR LA PILE

Entrée: A contient la moitié du nombre d'octet à réserver.

Sortie: Y inchangé.

Principe: on ajoute #3E=62,pour ne pas empiéter sur le tampon décimal,et pour laisser la place pour les IRQ.
62 octets semblent excessifs, mais communs à tous les BASIC Microsoft, il y a certainement une raison majeure.

```
C43B  ASL A       C437  ASL A       on veut 2*A octets sur la pile
C43C  ADC #3E     C438  ADC #3E     tout en réservant de la place pour les IRQ
C43E  BCS C483    C43A  BCS C47C    OUT OF MEMORY si déja trop bas
C440  STA 91      C43C  STA 91      sauver la plus basse valeur possible de S
C442  TSX         C43E  TSX         prendre le pointeur de pile
C443  CPX 91      C43F  CPX 91      et vérifier au dessus
C445  BCC C483    C441  BCC C47C    et OUT OF MEMORY si en dessous.
C447  RTS         C443  RTS
```

VERIFIER AY EN DESSOUS DES CHAINES

Entrée: AY contient la valeur désirée.

Sortie: OUT OF MEMORY après une réorganisation éventuelle si AY trop haut.
La zone #C7-CF n'est pas affèctée
AY est conservé.

```
C448  CPY A3      C444  CPY A3      comparer poids fort
C44A  BCC C474    C446  BCC C470    si inférieur,il y a la place,on sort.
C44C  BNE C452    C448  BNE C44E    si trop haut,réorganiser
C44E  CMP A2      C44A  CMP A2      comparer poids faible
C450  BCC C474    C44C  BCC C470    et sortir si inférieur
C452  PHA         C44E  PHA         faire une réorganisation sauver A
C453  LDX #09     C44F  LDX #09     sauver la zone #C7-#CF
C455  TYA         C451  TYA         ainsi que Y sur la pile
C456  PHA         C452  PHA
C457  LDA C6,X    C453  LDA C6,X
C459  DEX         C455  DEX
C45A  BPL C456    C456  BPL C452
C45C  JSR $D595   C458  JSR $D650   effectuer la réorganisation des chaines
C45F  LDX #F7     C45B  LDX #F7     et récupererla zone #C7-#CF.
C461  PLA         C45D  PLA         Attention:#D0+#F7=#C7 et non #1C7
```

```
C462  STA D0,X    C45E  STA D0,X    ainsi l'ont décidé les concepteurs du 6502
C464  INX         C460  INX         cet effet modulo n'est valable qu'en page 0
C465  BMI C461    C461  BMI C45D
C467  PLA         C463  PLA
C468  TAY         C464  TAY         récupérer Y
C469  PLA         C465  PLA         puis A
C46A  CPY A3      C466  CPY A3      et effectuer à nouveau la comparaison
C46C  BCC C474    C468  BCC C470
C46E  BNE C483    C46A  BNE C47C    OUT OF MEMORY si toujours trop haut
C470  CMP A2      C46C  CMP A2
C472  BCS C483    C46E  BCS C47C    idem pour poids faible
C474  RTS         C470  RTS
```

POINT D'ENTREE DE L'INTERPRETEUR (RESET A CHAUD)

```
C475  LDA 02C0    C471  LDA 02C0    Indiquer mode Texte
C478  AND #FE     C474  AND #FE     (LSR 02C0/ASL 02C0 aurait été mieux !)
C47A  STA 02C0    C476  STA 02C0
C47D  LSR 02F1    ..............    imprimante hors service (inutile)
C480  JMP $C4B5   C479  JMP $C4A8   et saut à l'interpréteur

C483  LDX #4D     C47C  LDX #4D     OUT OF MEMORY ERROR
```

AFFICHER UN MESSAGE D'ERREUR

Entrée: X contient le déplacement dans la table du premier caractère du message.

Sortie: le message est affiché et la main est rendue à l'interpréteur.
C'est le seul point d'entrée qui réinitialise les piles

```
..............    C47E  JSR $C82F   Imprimante hors service
C485  LSR 2E      C481  LSR 2E      inhiber le Ctrl O
C487  LSR 02F1    ..............    imprimante hors service
C48A  LSR 02F2    ..............
C48D  LSR 02F4    ..............    mode TROFF
C490  JSR $CB9F   C483  JSR $CBF0   aller à la ligne
C493  JSR $CC10   C486  JSR $CCD7   afficher un '?'
C496  LDA C2AC,X  C489  LDA C2A8,X  prendre le caractère du message
C499  PHA         C48C  PHA         sauver b7 surtout
C49A  AND #7F     C48D  AND #7F     éliminer b7
```

```
C49C  JSR $CC12   C48F  JSR $CCD9   et afficher
C49F  INX         C492  INX         préparer pour caractère suivant
C4A0  PLA         C493  PLA         récuperer le signe
C4A1  BPL C496    C494  BPL C489    et continuer si ce n'était pas le dernier
C4A3  JSR $C751   C496  JSR $C726   initialiser pile 6502 et descripteur
C4A6  LDA #AA     C499  LDA #A6
C4A8  LDY #C3     C49B  LDY #C3     AY pointe sur 'ERROR'
C4AA  JSR $CBED   C49D  JSR $CCB0   et afficher le message
C4AD  LDY A9      C4A0  LDY A9      prendre indicateur de mode direct
C4AF  INY         C4A2  INY
C4B0  BEQ C4B5    C4A3  BEQ C4A8    et sauter si mode direct
C4B2  JSR $E0B6   C4A5  JSR $E0BA   afficher IN xxxxx sinon
```

Bogue: sur la V1.1,le message 'Ready' est affiché avant de mettre hors service l'imprimante,on ne sort donc jamais d'un LPRINT malencontreux.

```
C4B5  JSR $CC8F   ..............  mode TROFF
..............    C4A8  LSR 0252   indiquer pas de IF encore
C4B8  LSR 2E      C4AB  LSR 2E     inhiber le Ctrl O
C4BA  LSR 02F1    ..............  imprimante hors service
C4BD  LSR 02F2    C4AD  LSR 02F2   indiquer retour interpréteur (LIST)
C4C0  LDA #B6     C4B0  LDA #B2
C4C2  LDY #C3     C4B2  LDY #C3    AY pointe sur 'Ready'
C4C4  JSR $001A   C4B4  JSR $001A  et afficher le message (si pas détourné)
C4C7  LSR 02F1    C4B7  JSR $C82F  imprimante hors service
C4CA  JSR $C5A2   C4BA  JSR $C592  saisir un ordre dans le tampon clavier
C4CD  STX E9      C4BD  STX E9     et ajuster TXTPTR
C4CF  STY EA      C4BF  STY EA     à 0034
C4D1  JSR $00E2   C4C1  JSR $00E2  prendre premier caractère du tampon
C4D4  TAX         C4C4  TAX        est-ce 00 (tampon vide)
C4D5  BEQ C4C7    C4C5  BEQ C4B7   oui,recommencer la saisie
C4D7  LDX #FF     C4C7  LDX #FF    si le tampon n'est pas vide (y compris ':')
C4D9  STX A9      C4C9  STX A9     indiquer mode direct
C4DB  BCC C4E3    C4CB  BCC C4D3   si c'est un chiffre,inserer la ligne
C4DD  JSR $C60A   C4CD  JSR $C5FA  coder le texte
C4E0  JMP $C8DD   C4D0  JMP $C90C  et saut à l'éxécution.
```

TRAITEMENT D'UNE LIGNE

```
C4E3  JSR $CA98   C4D3  JSR $CAE2  évaluer le numéro de ligne en $33-4
C4E6  JSR $C60A   C4D6  JSR $C5FA  coder le tampon
C4E9  STY 26      C4D9  STY 26     et sauver la longueur de la ligne
C4EB  JSR $C6DE   C4DB  JSR $C6B3  rechercher la ligne
C4EE  BCC C534    C4DE  BCC C524   si elle n'existe pas, insérer directement
```

Suppression d'une ligne

Principe: la routine est très optimisée et difficile à suivre !.

La plupart du travail consiste à ajuster correctement les pointeurs de début de bloc (en #91-#92),d'adresse cible (#93-#94),ainsi que la longueur du bloc à décaler: nombre de pages dans X,et fraction de page (en complément à 2) dans Y.

Le tout en ajustant les pointeurs source et cible pour pouvoir finir par décaler un nombre entier de page,de la même manière que #C3F8/#C3F4.

Le bloc est décalé en partant du bas,pour éviter d'écrire sur le bloc qu'on est en train de décaler.

Remarque:Cette routine n'est malheureusement pas exploitable car on retourne directement à l'interpréteur, sans passer par 'Ready' par exemple qui permettrait d'intercepter le retour.

```
C4F0  LDY #01     C4E0  LDY #01
C4F2  LDA (CE),Y  C4E2  LDA (CE),Y  poids fort adresse ligne suivante
C4F4  STA 92      C4E4  STA 92      (c'est le bas du bloc à décaler) en #92
C4F6  LDA 9C      C4E6  LDA 9C      poids faible fin du programme
C4F8  STA 91      C4E8  STA 91      en #91 (haut du bloc à décaler)
C4FA  LDA CF      C4EA  LDA CF      poids fort de l'adresse de la ligne
C4FC  STA 94      C4EC  STA 94      (c'est l'adresse 'cible' en #94
C4FE  LDA CE      C4EE  LDA CE      poids faible de l'adresse de la ligne
C500  DEY         C4F0  DEY         Y=0,C=0,soustraction inverse:complément à 2
C501  SBC (CE),Y  C4F1  SBC (CE),Y  moins adresse suivante=taille du 'trou'
C503  CLC         C4F3  CLC         ajouter à la fin du Basic
C504  ADC 9C      C4F4  ADC 9C      (en fait soustraire puisque complément à 2)
C506  STA 9C      C4F6  STA 9C      et ajuster fin du texte Basic
C508  STA 93      C4F8  STA 93      et sauver poids faible
C50A  LDA 9D      C4FA  LDA 9D      continuer l'addition en complément à 2
C50C  ADC #FF     C4FC  ADC #FF     donc la soustraction
C50E  STA 9D      C4FE  STA 9D      pour le poids fort de la fin du BASIC
C510  SBC CF      C500  SBC CF      calcul du nombre de page du bloc à décaler
C512  TAX         C502  TAX         dans X,compteur de pages
C513  SEC         C503  SEC         et idem pour le poids faible de la longueur
C514  LDA CE      C504  LDA CE      du bloc à décaler
C516  SBC 9C      C506  SBC 9C
C518  TAY         C508  TAY         en Y
C519  BCS C51E    C509  BCS C50E    et répercution éventuelle
C51B  INX         C50B  INX         sur le poids fort
C51C  DEC 94      C50C  DEC 94
C51E  CLC         C50E  CLC
C51F  ADC 91      C50F  ADC 91
```

```
C521  BCC C526    C511  BCC C516
C523  DEC 92      C513  DEC 92      et idem pour le pointeur 'source'
C525  CLC         C515  CLC
C526  LDA (91),Y  C516  LDA (91),Y  prendre l'octet
C528  STA (93),Y  C518  STA (93),Y  et le décaler vers sa nouvelle adresse
C52A  INY         C51A  INY
C52B  BNE C526    C51B  BNE C516    et continuer jusqu'à la fin de la page
C52D  INC 92      C51D  INC 92      préparer pour la page suivante
C52F  INC 94      C51F  INC 94
C531  DEX         C521  DEX         et tester si toutes les pages sont décalées
C532  BNE C526    C522  BNE C516
```

Insertion d'une ligne

Principe: passer correctement les paramètres à la routine de transfert.

Remarque: la ligne est transférée à partir de #31, c'est à dire lien poids faible en #31 (quelconque), et un lien poids fort en #32 qui ne doit pas être nul, sous peine d'éliminer la fin du programme.

```
C534  JSR $C733   C524  JSR $C708   faire CLEAR
C537  JSR $C56F   C527  JSR $C55F   restaurer les liens
C53A  LDA 35      C52A  LDA 35
C53C  BEQ C4C7    C52C  BEQ C4B7    si la ligne est vide,c'est fini
C53E  CLC         C52E  CLC
C53F  LDA 9C      C52F  LDA 9C      fin du BASIC
C541  STA C9      C531  STA C9      comme haut de la zone à décaler
C543  ADC 26      C533  ADC 26      plus longueur de la ligne
C545  STA C7      C535  STA C7      comme adresse cible
C547  LDY 9D      C537  LDY 9D      et idem pour les poids forts.
C549  STY CA      C539  STY CA
C54B  BCC C54E    C53B  BCC C53E
C54D  INY         C53D  INY
C54E  STY C8      C53E  STY C8
C550  JSR $C3F8   C540  JSR $C3F4   décaler le bloc
C553  LDA A0      C543  LDA A0      récuperer l'adresse cible
C555  LDY A1      C545  LDY A1      (sauvée par la routine #C3F8/#C3F4)
C557  STA 9C      C547  STA 9C
C559  STY 9D      C549  STY 9D      et ajuster pointeur de fin du BASIC
C55B  LDY 26      C54B  LDY 26      et enfin placer la ligne à sa place
C55D  DEY         C54D  DEY         y compris numéro (#33-#34) et un lien
C55E  LDA 0031,Y  C54E  LDA 0031,Y  non nul car #32 n'est jamais nul
C561  STA (CE),Y  C551  STA (CE),Y
C563  DEY         C553  DEY         décaler jusqu'à la fin
```

```
C564  BPL C55E     C554  BPL C54E     heureusement que le tampon fait moins de
C566  JSR $C733    C556  JSR $C708    128 octets ! faire CLEAR
C569  JSR $C56F    C559  JSR $C55F    restaurer les liens
C56C  JMP $C4C7    C55C  JMP $C4B7    et recommencer l'entrée d'un ordre.
```

RESTAURER LES LIENS DES LIGNES

Entrée: rien de spécial, #9A-B doit pointer sur le début du texte BASIC

Sortie: le lien est restauré, #91-2 pointe sur la fin du BASIC -2.

```
C56F  LDA 9A       C55F  LDA 9A
C571  LDY 9B       C561  LDY 9B       prendre début BASIC
C573  STA 91       C563  STA 91       comme pointeur de travail
C575  STY 92       C565  STY 92
C577  CLC          C567  CLC
C578  LDY #01      C568  LDY #01
C57A  LDA (91),Y   C56A  LDA (91),Y   prendre poids fort du lien
C57C  BEQ C59B     C56C  BEQ C58B     s'il est nul,c'est la fin du programme
C57E  LDY #04      C56E  LDY #04      indexer le premier caractère de la ligne
C580  INY          C570  INY          décrire toute la ligne
C581  LDA (91),Y   C571  LDA (91),Y
C583  BNE C580     C573  BNE C570     et continuer jusqu'à la fin de la ligne
C585  INY          C575  INY          pointer après le 0 de la ligne suivante
C586  TYA          C576  TYA          et calculer l'adresse de la ligne suivante
C587  ADC 91       C577  ADC 91
C589  TAX          C579  TAX          poids faible dans X
C58A  LDY #00      C57A  LDY #00
C58C  STA (91),Y   C57C  STA (91),Y   et le placer comme lien ligne courante
C58E  LDA 92       C57E  LDA 92       calculer maintenant le poids fort
C590  ADC #00      C580  ADC #00
C592  INY          C582  INY
C593  STA (91),Y   C583  STA (91),Y   et le sauver comme lien
C595  STX 91       C585  STX 91       et maintenant cela devient la
C597  STA 92       C587  STA 92       nouvelle ligne courante
C599  BCC C578     C589  BCC C568     inconditionnel:recommencer
C59B  RTS          C58B  RTS
```

PRENDRE UN ORDRE DANS LE TAMPON CLAVIER

Entrée: le point d'entrée est en #C5A2/#C592

Sortie: l'ordre entré est dans le tampon,terminé par un Ø.
XY vaut ØØ34

Remarque: c'est la seule routine qui traite le Ctrl A.

Principe: X compte les caractères entrés.
Le Ctrl A est traité de manière originale,de telle sorte qu'on se retrouve dans les mêmes conditions que si le caractère est directement frappé. C'est un principe général (ramener les cas particuliers au cas général) qui permet de gagner très souvent de la place.

```
C59C  DEX          C58C  DEX          DEL:décrémenter le pointeur
C59D  BPL C5A4     C58D  BPL C594     et continuer si le tampon n'est pas vide
C59F  JSR $CB9F    C58F  JSR $CBFØ    si le tampon est vide,aller à la ligne
```

Saisie d'un ordre: point d'entrée

```
C5A2  LDX #ØØ      C592  LDX #ØØ      initialiser le pointeur
C5A4  JSR $C5F8    C594  JSR $C5E8    et prendre un caractère au clavier
C5A7  CMP #Ø1      C597  CMP #Ø1      tester pour Ctrl A
C5A9  BNE C5B8     C599  BNE C5A8     non,sauter
C5AB  LDY Ø269     C59B  LDY Ø269     Ctrl A:prendre la colonne du curseur
C5AE  LDA (12),Y   C59E  LDA (12),Y   et le caractère sous le curseur
C5BØ  AND #7F      C5AØ  AND #7F      éliminer la video inverse du clignotement
C5B2  CMP #2Ø      C5A2  CMP #2Ø      est-ce un attribut vidéo ?
C5B4  BCS C5B8     C5A4  BCS C5A8
C5B6  LDA #Ø9      C5A6  LDA #Ø9      oui,l'ignorer simplement
C5B8  PHA          C5A8  PHA          sauver le caractère frappé
C5B9  JSR $CC12    C5A9  JSR $CCD9    et l'afficher
C5BC  PLA          C5AC  PLA          le récuperer
C5BD  CMP #7F      C5AD  CMP #7F      est-ce DEL ?
C5BF  BEQ C59C     C5AF  BEQ C58C     oui,l'exécuter
C5C1  CMP #ØD      C5B1  CMP #ØD      est-ce Return ?
C5C3  BEQ C5F5     C5B3  BEQ C5E5     oui,sortir
C5C5  CMP #Ø3      C5B5  CMP #Ø3      est-ce Ctrl C ?
C5C7  BEQ C5F1     C5B7  BEQ C5E1     oui,annuler et sortir
C5C9  CMP #18      C5B9  CMP #18      est-ce Ctrl X ?
C5CB  BEQ C5D8     C5BB  BEQ C5C8     oui,annuler et recommencer
C5CD  CMP #2Ø      C5BD  CMP #2Ø      était-ce un caractère de controle ?
C5CF  BCC C5A4     C5BF  BCC C594     oui ,continuer
C5D1  STA 35,X     C5C1  STA 35,X     non,le stocker dans le tampon
C5D3  INX          C5C3  INX          et incrémenter le pointeur
C5D4  CPX #4F      C5C4  CPX #4F      était-ce le 79ème caractère ?
```

```
C5D6  BCC C5DF    C5C6  BCC C5CF    non,continuer

C5D8  LDA #' '    C5C8  LDA #' '    Ctrl X: afficher un ' '
C5DA  JSR $CC12   C5CA  JSR $CCD9
C5DD  BNE C59F    C5CD  BNE C58F    puis aller à la ligne et recommencer

C5DF  CPX #4C     C5CF  CPX #4C     était-ce au moins le 76éme caractère ?
C5E1  BCC C5A4    C5D1  BCC C594     non,continuer
C5E3  TXA         C5D3  TXA         sauver le pointeur de tampon
C5E4  PHA         C5D4  PHA         sur la pile
C5E5  TYA         C5D5  TYA         et Y (inutile !)
C5E6  PHA         C5D6  PHA
C5E7  JSR $F412   C5D7  JSR $FA9F   et envoyer un PING
C5EA  PLA         C5DA  PLA
C5EB  TAY         C5DB  TAY         récupérer Y
C5EC  PLA         C5DC  PLA
C5ED  TAX         C5DD  TAX         et X
C5EE  JMP $C5A4   C5DE  JMP $C594   et continuer

C5F1  INC 17      C5E1  INC 17      Ctrl C:placer indicateur pour INPUT
C5F3  LDX #00     C5E3  LDX #00
C5F5  JMP $CB99   C5E5  JMP $CBEA   Return:placer un #00 et sortir
```

SAISIR UN CARACTERE AU CLAVIER

Entrée: rien de spécial

Sortie: A contient le caractère frappé,X et Y sont conservés.

Remarque: une vectorisation a été rajoutée sur la V1.1,très pratique pour ajouter des touches de fonction par exemple.
C'est la seule routine qui traite le Ctrl O.

```
C5F8  JSR $E905   C5E8  JSR $023B   prendre le tampon de touche
C5FB  BPL C5F8    C5EB  BPL C5E8    s'il est vide,recommencer
C5FD  CMP #0F     C5ED  CMP #0F     est-ce Ctrl O ?
C5FF  BNE C609    C5EF  BNE C5F9    non,sortir
C601  PHA         C5F1  PHA         oui,le sauver
C602  LDA 2E      C5F2  LDA 2E      et inverser l'indicateur
C604  EOR #FF     C5F4  EOR #FF
C606  STA 2E      C5F6  STA 2E      puis le resauver
C608  PLA         C5F8  PLA         et récupérer le code du Ctrl O
C609  RTS         C5F9  RTS
```

ENCODAGE DES MOTS CLES

Entrée: #E9 pointe sur le premier caractère à analyser. L'encodage se fait dans le tampon clavier, et le texte à coder doit être terminé par un Ø.

Sortie: Y pointe sur le dernier caractère de la commande codée.

Le texte codé se retrouve aussi dans le tampon clavier, terminé par un Ø

Un #ØØ est aussi mis 2 caractères plus loin pour simuler la fin d'un programme (Poids fort du lien des lignes nul), lors de l'éxécution d'un ordre en mode direct, ce qui permet de traiter le mode direct comme le mode programme.

Principe: X pointe la commande non codée, et Y la commande codée. On a donc toujours X >= Y car le codage ne peut que raccourcir l'entrée.

Des précautions sont prises pour ne pas coder les chaines entre guillemets, les DATA et les REM.

Les chaines et les REM sont recopiées intégralement à part, pour les DATA, c'est la variable #2A qui dit si on doit ou non coder (b6=Ø si on doit coder)

La routine, pour gagner du temps (?), n'essaye pas de coder les chiffres

Dans les autres cas, elle procède par essais successifs, essayant tous les mots-clés jusqu'à ce qu'un convienne. En dernier ressort, le caractère est recopié tel quel.

Cette méthode explique le fait que la routine soit longue à exécuter, ce qui n'est pas génant puis qu'elle n'est appelée qu'une fois,et jamais par l'exécution.

```
C60A  LDX E9      C5FA  LDX E9      initialiser le pointeur 'ancien'
C60C  LDY #04     C5FC  LDY #04     et le pointeur 'nouveau'
C60E  STY 2A      C5FE  STY 2A      et l'indicateur (b6=Ø)
C610  LDA ØØ,X    C600  LDA ØØ,X    prendre un caractère
C612  CMP #20     C602  CMP #20     est-ce un espace ?
C614  BEQ C657    C604  BEQ C647    oui,recopier simplement
C616  STA 25      C606  STA 25      non,sauver le code pour terminateur
C618  CMP #'"'    C608  CMP #'"'    est-ce un début (ou fin) de chaine ?
C61A  BEQ C67B    C60A  BEQ C66B    oui,recopier et traiter
C61C  BIT 2A      C60C  BIT 2A      peut-on coder ?
C61E  BVS C657    C60E  BVS C647    non,recopier simplement
C620  CMP #'?'    C610  CMP #'?'    est-ce PRINT abrégé ?
C622  BNE C628    C612  BNE C618    non,sauter
C624  LDA #BA     C614  LDA #BA     oui,alors token pour PRINT
C626  BNE C657    C616  BNE C647    Inconditionel:et le placer
```

```
C628  CMP #'Ø'     C618  CMP #'Ø'     si c'est un chiffre
C62A  BCC C630     C61A  BCC C620
C62C  CMP #3C      C61C  CMP #3C      ou ':' ou ';',inutile de coder
C62E  BCC C657     C61E  BCC C647     et recopier simplement
C630  STY E0       C620  STY E0       sauver Y
C632  LDY #00      C622  LDY #00
C634  STY 26       C624  STY 26       compteur de mot clé=Ø
C636  LDA #E9      C626  LDA #E9
C638  STA 18       C628  STA 18       et initialiser le pointeur
C63A  LDA #C0      C62A  LDA #C0      au début de la table des mots-clés
C63C  STA 19       C62C  STA 19       (-1 car on commence par incrémenter)
C63E  STX E9       C62E  STX E9       sauver début du mot à coder
C640  DEX          C630  DEX          et ajuster car on commence par incrémenter
C641  INX          C631  INX          caractère suivant
C642  INC 18       C632  INC 18       et idem pour
C644  BNE C648     C634  BNE C638
C646  INC 19       C636  INC 19       la table des mots-clés
C648  LDA 00,X     C638  LDA 00,X     prendre le caractère dans le tampon
C64A  SEC          C63A  SEC
C64B  SBC (18),Y   C63B  SBC (18),Y   et soustraire caractère mot-clé
C64D  BEQ C641     C63D  BEQ C631     si pareil,continuer même mot-clé
C64F  CMP #80      C63F  CMP #80      si c'est #8Ø, c'était le dernier caractère
C651  BNE C682     C641  BNE C672     sinon, passer au mot clé suivant
C653  ORA 26       C643  ORA 26       Le mot-clé est trouvé: b7=1
C655  LDY E0       C645  LDY E0       récuperer Y
C657  INX          C647  INX          X pointe après le mot trouvé
C658  INY          C648  INY          et Y sur la position courante
C659  STA 0030,Y   C649  STA 0030,Y   on sauve le code (ou le caractère)
C65C  LDA 0030,Y   C64C  LDA 0030,Y   était-ce Ø ?
C65F  BEQ C69D     C64F  BEQ C68A     oui, le codage est fini: sortir

C661  SEC          C651  SEC
C662  SBC #':'     C652  SBC #':'     est-ce la fin d'une instruction ?
C664  BEQ C66A     C654  BEQ C65A     oui, baisser indicateur DATA
C666  CMP #57      C656  CMP #57      est-ce DATA (#3A+#57=#91=token DATA)
C668  BNE C66C     C658  BNE C65C     non, sauter
C66A  STA 2A       C65A  STA 2A       placer drapeau DATA (b6=1 ou Ø)
C66C  SEC          C65C  SEC          tester le code pour REM
C66D  SBC #63      C65D  SBC #63      (#3A+#63=#9D=token REM)
C66F  BNE C610     C65F  BNE C600     si pas REM,continuer

Ignorer Chaine ou REM

C671  STA 25       C661  STA 25       Terminateur=Ø,ou " pour une chaine
```

```
C673  LDA ØØ,X    C663  LDA ØØ,X    prendre les caractères du tampon
C675  BEQ C657    C665  BEQ C647    si fin,recopier Ø et sortir
C677  CMP 25      C667  CMP 25      est-ce un terminateur ?
C679  BEQ C657    C669  BEQ C647    oui,le recopier et continuer
C67B  INY         C66B  INY         non,recopier le caractère
C67C  STA ØØ3Ø,Y  C66C  STA ØØ3Ø,Y
C67F  INX         C66F  INX         passer au caractère suivant
C68Ø  BNE C673    C67Ø  BNE C663    inconditionnel
```

Passer au mot clé suivant

```
C682  LDX E9      C672  LDX E9      récupérer le pointeur
C684  INC 26      C674  INC 26      et incrémenter le numéro du mot-clé
C686  LDA (18),Y  C676  LDA (18),Y  et rechercher la fin du mot-clé courant
C688  PHP         C678  PHP         sauver le signe (b7)
C689  INC 18      C679  INC 18      passer au caractère suivant
C68B  BNE C68F    C67B  BNE C67F
C68D  INC 19      C67D  INC 19      dans la table des mots-clé
C68F  PLP         C67F  PLP         récuperer le signe
C69Ø  BPL C686    C68Ø  BPL C676    et recommencer si ce n'était pas le dernier
C692  LDA (18),Y  C682  LDA (18),Y  prendre premier caractère du mot suivant
C694  BEQ C699    C684  BNE C638    si pas Ø (fin de la table),recommencer
C696  JMP $C648   ..............    3 octets gaspillés...
C699  LDA ØØ,X    C686  LDA ØØ,X    si pas mot-clé,récupérer le caractère
C69B  BPL C655    C688  BPL C645    Inconditionnel:et le recopier
```

Terminer

```
C69D  STA ØØ32,Y  C68A  STA ØØ32,Y  mettre un Ø 2 caractères plus loin
C6AØ  LDA #34     C68D  LDA #34     et ajuster TXTPTR sur le début de la ligne
C6A2  STA E9      C68F  STA E9
C6A4  RTS         C691  RTS
```

'EDIT' (COMMANDE)

Action: liste une ligne et essaye de mettre le curseur au début de la ligne listée. En fait fait à peu près n'importe quoi,ce qui le rend inutile.

Bogue: V1.Ø: le curseur est éteint pour éviter que le fait de changer directement le numéro de la ligne ne laisse des traces.Malheureusement, s'il était éteint,il sera rallumé...

D'autre part,le fait de sauver la ligne de départ ne sert à rien s'il y a un scroll entre temps.

V1.1: les auteurs de la ROM ont carrément estimé q'une ligne BASIC une fois listée tenait dans une ligne d'écran ! Dommage.

Une bogue commune: il n'est pas testé que le numéro de ligne est bien le seul paramètre.EDIT 10,AAAAAAA ne déclenche pas de message d'erreur !

```
C6A5  JSR $CA98   C692  JSR $CAE2   évaluer le numéro de ligne en #33-4
C6A8  JSR $C6DE   C695  JSR $C6B3   et chercher la ligne
C6AB  BCC C6DB    C698  BCC C6B0   si pas trouvée,erreur
C6AD  LDA #80     ..............   indiquer retour par RTS
C6AF  STA 02F2    ..............   du sous programme LIST
C6B2  LDA #11     ..............   (Ctrl Q)
C6B4  JSR $CC12   ..............   effacer le curseur (s'il était allumé)
C6B7  LDA 0268    ..............   sauver la ligne du curseur
C6BA  PHA         ..............   sur la pile
C6BB  NOP         ..............
C6BC  NOP         ..............
C6BD  NOP         ..............
C6BE  NOP         ..............
C6BF  NOP         ..............
C6C0  NOP         ..............
C6C1  JSR $C799   ..............   lister la ligne concernée
C6C4  PLA         ..............
C6C5  STA 0268    ..............   récupérer la ligne originale
C6C8  JSR $CB9F   ..............   et aller à la ligne (car l'édition
C6CB  NOP         ..............   d'une ligne commence aussi par aller
C6CC  NOP         ..............   à la ligne)
C6CD  NOP         ..............
C6CE  LDA #11     ..............   et enfin replacer le curseur
C6D0  JSR $CC12   ..............   dans sa couleur d'origine
C6D3  LSR 02F2    ..............   et remettre #2F2 normal (b7=0)
..............    C69A  ROR 02F2   C=1 donc b7=1:indiquer retour par RTS
..............    C69D  JSR $C76C  lister la ligne
..............    C6A0  LSR 02F2   réinitialiser #2F2 (b7=0)
..............    C6A3  JSR $CBF0  aller à la ligne
..............    C6A6  LDA #0B    et remonter d'une ligne
..............    C6A8  JSR $CCD9  pour se placer au début si une seule ligne
C6D6  PLA         C6AB  PLA        enlever l'adresse de retour
C6D7  PLA         C6AC  PLA
C6D8  JMP $C4C7   C6AD  JMP $C4B7  et retourner à l'interpréteur
C6DB  JMP $C9F1   C6B0  JMP $CA23  'UNDEF'D STATEMENT ERROR'
```

RECHERCHE D'UNE LIGNE BASIC

Entrée: #33-4 contient le numéro de la ligne à chercher.

Sortie: C=Ø si la ligne n'est pas trouvée (#CE-#CF pointe sur la ligne de numéro immédiatement supérieur, ou la fin du programme.)
C=1 si la ligne est trouvée,#CE-#CF contient alors l'adresse de la ligne (pointant sur le caractère suivant le Ø de début de ligne).

Principe: grâce aux liens entre les lignes,qui trouvent ici leur unique mais suffisante justification, la routine saute très vite d'une ligne à l'autre pour comparer les numéros.

Curiosité: à chaque ligne rencontrée, le pointeur #1D-#1E est incrémenté. Selon le point d'entrée, ce pointeur est ou non remis à zéro. Totalement inutile.

```
C6DE  LDA #ØØ      C6B3  LDA #ØØ      initialiser
C6EØ  STA 1D       C6B5  STA 1D
C6E2  STA 1E       C6B7  STA 1E       le compteur de lignes
C6E4  LDA 9A       C6B9  LDA 9A       prendre le début du Basic
C6E6  LDX 9B       C6BB  LDX 9B       comme début de recherche
C6E8  LDY #Ø1      C6BD  LDY #Ø1      Y indexe le poids fort du lien d'une ligne
C6EA  STA CE       C6BF  STA CE
C6EC  STX CF       C6C1  STX CF       placer adresse ligne courante
C6EE  LDA (CE),Y   C6C3  LDA (CE),Y   prendre lien poids fort
C6FØ  BEQ C717     C6C5  BEQ C6EC     si nul,fin du programme:ligne non trouvée
C6F2  INY          C6C7  INY          sinon,indexer le poids fort du numéro
C6F3  INY          C6C8  INY          (LDY #$Ø3 aurait fait gagner 2 micro sec.)
C6F4  INC 1D       C6C9  INC 1D       incrémenter le compteur de ligne
C6F6  BNE C6FA     C6CB  BNE C6CF     le supprimer aurait fait gagner 8 microsec.
C6F8  INC 1E       C6CD  INC 1E       ce qui est négligeable il est vrai...
C6FA  LDA 34       C6CF  LDA 34       prendre numéro de la ligne à trouver
C6FC  CMP (CE),Y   C6D1  CMP (CE),Y   et comparer
C6FE  BCC C718     C6D3  BCC C6ED     si déjà inferieur, ligne pas trouvée
C7ØØ  BEQ C7Ø5     C6D5  BEQ C6DA     si égal,comparer poids faible
C7Ø2  DEY          C6D7  DEY          continuer la recherche (ajuster Y)
C7Ø3  BNE C7ØE     C6D8  BNE C6E3     inconditionnel
C7Ø5  LDA 33       C6DA  LDA 33       prendre poids faible maintenant
C7Ø7  DEY          C6DC  DEY          ajuster Y
C7Ø8  CMP (CE),Y   C6DD  CMP (CE),Y   et comparer
C7ØA  BCC C718     C6DF  BCC C6ED     si déjà inférieur, ligne pas trouvée,C=Ø
C7ØC  BEQ C718     C6E1  BEQ C6ED     si égal,ligne trouvée et C=1
C7ØE  DEY          C6E3  DEY          passer à la ligne suivante
C7ØF  LDA (CE),Y   C6E4  LDA (CE),Y   prendre lien poids fort
```

```
C711  TAX          C6E6  TAX          le sauver dans X
C712  DEY          C6E7  DEY
C713  LDA (CE),Y   C6E8  LDA (CE),Y   puis lien poids faible dans A
C715  BCS C6E8     C6EA  BCS C6BD     inconditionnel:continuer la recherche
C717  CLC          C6EC  CLC          indiquer ligne pas trouvée
C718  RTS          C6ED  RTS
```

'NEW' (COMMANDE)

Action: mettre deux zéro à la place du lien de la première ligne et effectuer un CLEAR, puis repartir à l'interpréteur.

```
C719  BNE C718     C6EE  BNE C6ED     s'il y a des paramètres,retour
C71B  LDA #00      C6F0  LDA #00
C71D  LSR 02F4     C6F2  LSR 02F4
C720  TAY          C6F5  TAY          Y=0
C721  STA (9A),Y   C6F6  STA (9A),Y   annuler poids faible lien (inutile)
C723  INY          C6F8  INY
C724  STA (9A),Y   C6F9  STA (9A),Y   et annuler poids fort lien (indispensable)
C726  LDA 9A       C6FB  LDA 9A       ajuster le pointeur
C728  CLC          C6FD  CLC
C729  ADC #02      C6FE  ADC #02      de fin de programme
C72B  STA 9C       C700  STA 9C       poids faible
C72D  LDA 9B       C702  LDA 9B
C72F  ADC #00      C704  ADC #00
C731  STA 9D       C706  STA 9D       idem poids fort
C733  JSR $C765    C708  JSR $C73A    placer TXTPTR au début du programme
C736  LDA #00      C70B  LDA #00      Z=1 (BYT #2C aurait fait gagner un octet)
```

'CLEAR' (COMMANDE)

Action: initialise tous les pointeurs de variables et éxécute un RESTORE.
La pile 6502 et la pile des descripteurs est aussi initialisée.
Seule l'adresse de retour est conservée sur la pile

Remarque: la pile 6502 est initialisée à #FE et non à #FF,pour laisser à son sommet le #00 mis lors de l'initialisation, qui sert de butée à la routine de recherche d'un bloc FOR.

```
C738  BNE C764     C70D  BNE C739     retour s'il y a des paramètres
C73A  LDA A6       C70F  LDA A6
C73C  LDY A7       C711  LDY A7       placer HIMEM
```

```
C73E  STA A2      C713  STA A2
C740  STY A3      C715  STY A3     comme bas des chaines
C742  LDA 9C      C717  LDA 9C
C744  LDY 9D      C719  LDY 9D     prendre fin du BASIC
C746  STA 9E      C71B  STA 9E
C748  STY 9F      C71D  STY 9F     comme fin des variables
C74A  STA A0      C71F  STA A0
C74C  STY A1      C721  STY A1     et des tableaux
C74E  JSR $C91F   C723  JSR $C952  faire RESTORE
C751  LDX #88     C726  LDX #88    initialiser
C753  STX 85      C728  STX 85     la pile des descripteurs
C755  PLA         C72A  PLA        récupérer l'adresse de retour
C756  TAY         C72B  TAY
C757  PLA         C72C  PLA        dans AY
C758  LDX #FE     C72D  LDX #FE    initialiser
C75A  TXS         C72F  TXS        la pile 6502
C75B  PHA         C730  PHA
C75C  TYA         C731  TYA
C75D  PHA         C732  PHA        et resauver l'adresse de retour
C75E  LDA #00     C733  LDA #00
C760  STA AD      C735  STA AD     indiquer CONT impossible
C762  STA 2B      C737  STA 2B     et initialiser drapeau variables permises
C764  RTS         C739  RTS
```

PLACER TXTPTR AU DEBUT DU PROGRAMME BASIC

Principe: curieusement, le principe de l'addition a été retenu pour soustraire un au contenu du pointeur #9A. Le principe du complément à 2 a connu des utilisations plus optimales. Donc on ajoute #FFFF au lieu de retrancher 1.

```
C765  CLC         C73A  CLC
C766  LDA 9A      C73B  LDA 9A     prendre poids faible début Basic
C768  ADC #FF     C73D  ADC #FF    et ajouter -1
C76A  STA E9      C73F  STA E9     le sauver à TXTPTR
C76C  LDA 9B      C741  LDA 9B
C76E  ADC #FF     C743  ADC #FF
C770  STA EA      C745  STA EA     et idem poids fort.
C772  RTS         C747  RTS
```

'LIST' (COMMANDE)

Principe: le listage des mots-clés est un principe très classique, qu'il est inutile de développer.

Remarque: si on appuie sur une touche pendant le listing, #02DF ne contiendra pas #00 à la sortie, car on se contente de LSR 02DF,ce qui est suffisant.

```
C773 PHP          C748 PHP          sauver P (test des paramètres)
C774 JSR $CA98    C749 JSR $CAE2    évaluer un No de ligne en #33-4 (0 sinon)
C777 JSR $C6DE    C74C JSR $C6B3    et chercher la ligne
C77A PLP          C74F PLP          au fait,y-a-t-il des paramètres ?
C77B BEQ C791     C750 BEQ C766     non,sauter
C77D JSR $00E8    C752 JSR $00E8    oui,prendre caractère courant
C780 BEQ C797     C755 BEQ C76C     si vide c'était un numéro de ligne seul
C782 CMP #CD      C757 CMP #CD      est-ce '-' (token Basic)
C784 BNE C718     C759 BNE C6ED     non,sortir prématurément
C786 JSR $00E2    C75B JSR $00E2    oui,le sauter
C789 BEQ C791     C75E BEQ C766     c'est tout ?
C78B JSR $CA98    C760 JSR $CAE2    non,évaluer un autre numéro de ligne
C78E BEQ C797     C763 BEQ C76C     et continuer
C790 RTS          C765 RTS          sortir si encore des paramétres

C791 LDA #FF      C766 LDA #FF      initialiser dernière
C793 STA 33       C768 STA 33       ligne la plus haute si pas précisée
C795 STA 34       C76A STA 34
C797 NOP          ..............
C798 NOP          ..............
C799 LDY #01      C76C LDY #01      indexer le lien poids fort
C79B LDA (CE),Y   C76E LDA (CE),Y
C79D BEQ C7E6     C770 BEQ C7BF     si nul,fin programme et de la liste:sortir
C79F JSR $C930    C772 JSR $C962    tester clavier et Ctrl C
C7A2 CMP #' '     C775 CMP #' '     si pas espace,
C7A4 BNE C7AE     C777 BNE C787     continuer
C7A6 LSR 02DF     C779 LSR 02DF     viser tampon touche
C7A9 JSR $E905    ..............    attendre une touche
C7AC BPL C7A9     ..............
.............     C77C LDA 02DF
.............     C77F BPL C77C     attendre une touche
.............     C781 JSR $C962    tester si Ctrl C
.............     C784 LSR 02DF     non,vider tampon
C7AE INY          C787 INY          indexer le numéro de ligne
C7AF LDA (CE),Y   C788 LDA (CE),Y   le charger (poids faible)
C7B1 TAX          C78A TAX          dans X aussi
C7B2 INY          C78B INY          indexer poids fort
C7B3 LDA (CE),Y   C78C LDA (CE),Y   le charger
C7B5 CMP 34       C78E CMP 34       et comparer à valeur limite
C7B7 BNE C7BD     C790 BNE C796     si pas égal tester si pas trop grand
C7B9 CPX 33       C792 CPX 33       test poids faible aussi
```

```
C7BB  BEQ C7BF    C794  BEQ C798    si égal,traiter quand même
C7BD  BCS C7E6    C796  BCS C7BF    si dernière ligne dépassée,sortir
C7BF  STY B8      C798  STY B8      sauver pointeur de ligne
C7C1  PHA         C79A  PHA         et numéro poids fort
C7C2  JSR $CB9F   C79B  JSR $CBFØ   aller à la ligne (ne touche pas à X)
C7C5  PLA         C79E  PLA         récuperer numéro poids fort
C7C6  JSR $EØC1   C79F  JSR $EØC5   et afficher le numéro de ligne
C7C9  LDA #2Ø     C7A2  LDA #2Ø     commencer par afficher un espace
C7CB  LDY B8      C7A4  LDY B8      récupérer l'index de ligne
C7CD  AND #7F     C7A6  AND #7F     éliminer b7 (dernière lettre mot-clé)
C7CF  JSR $CC12   C7A8  JSR $CCD9   afficher un caractère
C7D2  INY         C7AB  INY         passer au caractère suivant
C7D3  BEQ C7E6    C7AC  BEQ C7BF    si la ligne est trop longue,sortir
C7D5  LDA (CE),Y  C7AE  LDA (CE),Y  prendre le caractère
C7D7  BNE C7F7    C7BØ  BNE C7DØ    aller le traiter
C7D9  TAY         C7B2  TAY         si fin de la ligne,Y=Ø
C7DA  LDA (CE),Y  C7B3  LDA (CE),Y  prendre lien poids faible
C7DC  TAX         C7B5  TAX         dans X
C7DD  INY         C7B6  INY         indexer poids fort
C7DE  LDA (CE),Y  C7B7  LDA (CE),Y  prendre lien poids fort
C7EØ  STX CE      C7B9  STX CE      et initialiser adresse
C7E2  STA CF      C7BB  STA CF      de la ligne suivante
C7E4  BNE C799    C7BD  BNE C76C    et continuer si pas fin du programme
C7E6  BIT Ø2F2    C7BF  BIT Ø2F2    test du mode de retour
C7E9  BPL C7EC    C7C2  BPL C7C5    b7=Ø:retour interpréteur
C7EB  RTS         C7C4  RTS         sinon simple retour
C7EC  JSR $CB9F   C7C5  JSR $CBFØ   aller à la ligne
C7EF  LSR Ø2F1    C7C8  JSR $C82F   imprimante hors service
C7F2  PLA         C7CB  PLA         enlever adresse de retour à l'interpréteur
C7F3  PLA         C7CC  PLA         pour ajuster la pile
C7F4  JMP $C4B5   C7CD  JMP $C4A8   et saut à l'interpréteur
```

Lister un caractère

```
C7F7  BPL C7CF    C7DØ  BPL C7A8    si positif,afficher simplement
C7F9  SEC         C7D2  SEC
C7FA  SBC #7F     C7D3  SBC #7F     sinon ramener au numéro du mot-clé
C7FC  TAX         C7D5  TAX         comme index
C7FD  STY B8      C7D6  STY B8      sauver l'index de ligne
C7FF  LDY #ØØ     C7D8  LDY #ØØ     et initialiser l'index des mots-clés
C8Ø1  LDA #E9     C7DA  LDA #E9
C8Ø3  STA 18      C7DC  STA 18
C8Ø5  LDA #CØ     C7DE  LDA #CØ
C8Ø7  STA 19      C7EØ  STA 19      ainsi que l'adresse de la table des mots.
```

```
C8Ø9  DEX          C7E2  DEX          passer mot-clé suivant
C8ØA  BEQ C819     C7E3  BEQ C7F2     si c'est le bon,l'afficher
C8ØC  INC 18       C7E5  INC 18       incrémenter le pointeur
C8ØE  BNE C812     C7E7  BNE C7EB
C81Ø  INC 19       C7E9  INC 19       des mots-clés
C812  LDA (18),Y   C7EB  LDA (18),Y   et prendre un caractère
C814  BPL C8ØC     C7ED  BPL C7E5     si b7=Ø,continuer
C816  JMP $C8Ø9    C7EF  JMP $C7E2    passer mot-clé suivant.BMI serait judicieux
C819  INY          C7F2  INY          indexer le premier caractére puis suivants
C81A  LDA (18),Y   C7F3  LDA (18),Y   prendre caractère mot-clé
C81C  BMI C7CB     C7F5  BMI C7A4     b7=1,c'est le dernier:retour
C81E  JSR $CC12    C7F7  JSR $CCD9    c'est pas le dernier,on l'affiche
C821  JMP $C819    C7FA  JMP $C7F2    et on continue
```

'LLIST' (COMMANDE)

Remarque:l'imprimante est remise hors service par la routine LIST elle même dans le cas où on ne sort pas par RTS.

```
C824  LDA #8Ø      ..............  mettre drapeau imprimante
C826  STA Ø2F1     ..............   (SEC:ROR serait plus judicieux)
C829  LSR Ø2F2     ..............  indiquer pas retour par RTS
C82C  JSR $ØØE8    ..............  replacer les indicateurs
C82F  JMP $C773    ..............  et faire LIST
..............  C7FD  JSR $C816  mettre l'imprimante en service
..............  C8ØØ  LSR Ø2F2   indiquer pas retour par RTS
..............  C8Ø3  JSR $ØØE8  replacer les indicateurs
..............  C8Ø6  JMP $C748  et faire LIST
```

'LPRINT' (COMMANDE)

```
C832  LDA #8Ø      ..............
C834  STA Ø2F1     ..............  Mettre drapeau imprimante
C837  JSR $ØØE8    ..............  replacer les indicateurs
C83A  JSR $CB61    ..............  faire PRINT
C83D  LSR Ø2F1     ..............  et remettre l'imprimante hors service
C84Ø  RTS          ..............
..............  C8Ø9  JSR $C816  mettre l'imprimante en service
..............  C8Ø C  JSR $ØØE8  replacer les indicateurs
..............  C8ØF  JSR $CBAB  faire 'PRINT'
..............  C812  JSR $C82F  et remettre l'imprimante hors service
..............  C815  RTS        JMP $C82F aurait été aussi bien
```

METTRE L'IMPRIMANTE EN SERVICE

```
..... C816 BIT Ø2F1    si déjà en service,sortir
..... C819 BMI C854
..... C81B LDA 3Ø      sauver position curseur écran
..... C81D STA Ø259
..... C82Ø LDA Ø258    et récuperer
..... C823 STA 3Ø      la position pour l'imprimante
..... C825 SEC
..... C826 ROR Ø2F1
..... C829 LDA Ø256    prendre longueur d'une ligne d'impression
..... C82C JMP $C844   et finir
```

METTRE L'IMPRIMANTE HORS SERVICE

```
..... C82F BIT Ø2F1    si déjà hors service,sortir
..... C832 BPL C854
..... C834 LDA 3Ø      sauver position curseur imprimante
..... C836 STA Ø258
..... C839 LDA Ø259    et récupérer position
..... C83C STA 3Ø      curseur écran
..... C83E LSR Ø2F1    placer indicateur
..... C841 LDA Ø257    prendre longueur ligne d'écran
..... C844 STA 31      et la sauver comme longueur courante
..... C846 SEC         calculer la valeur de la longueur
..... C847 SBC #Ø8     modulo 8 entre #F8 et #FF
..... C849 BCS C846
..... C84B EOR #FF     ramener á Ø à 7 (complément à 2)
..... C84D SBC #Ø6     et soustraire 7 (C=Ø):#F9 à #ØØ
..... C84F CLC         et enfin ajouter à
..... C85Ø ADC 31      la longueur totale permise
..... C852 STA 32      on obtient la position maxi de tabulation
..... C854 RTS
```

'FOR' (COMMANDE)

```
C841 LDA #Ø8     C855 LDA #Ø8     Indiquer pas d'entier
C843 STA 2B      C857 STA 2B
C845 JSR $CAD2   C859 JSR $CB1C   faire LET pour l'indice
C848 JSR $C3CA   C85C JSR $C3C6   chercher si déjà une boucle avec cet indice
C84B BNE C852    C85F BNE C866    non,sauter
C84D TXA         C861 TXA         oui,ajuster pointeur de pile
C84E ADC #ØF     C862 ADC #ØF     pour sauter ce bloc
```

```
C850  TAX          C864  TAX
C851  TXS          C865  TXS
C852  PLA          C866  PLA          enlever l'adresse de retour
C853  PLA          C867  PLA          à l'intepréteur
C854  LDA #09      C868  LDA #09      demander 18 octets sur la pile
C856  JSR $C43B    C86A  JSR $C437
C859  JSR $CA1C    C86D  JSR $CA4E    chercher la fin de l'instruction
C85C  CLC          C870  CLC          (le résultat est dans Y)
C85D  TYA          C871  TYA
C85E  ADC E9       C872  ADC E9       et l'ajouter à TXTPTR
C860  PHA          C874  PHA          sauver poids faible sur la pile
C861  LDA EA       C875  LDA EA       idem poids fort
C863  ADC #00      C877  ADC #00      on empile donc l'adresse du début effectif
C865  PHA          C879  PHA          de la boucle
C866  LDA A9       C87A  LDA A9       empiler aussi le numéro de la ligne
C868  PHA          C87C  PHA          poids fort
C869  LDA A8       C87D  LDA A8       et idem poids faible
C86B  PHA          C87F  PHA
C86C  LDA #&T0     C880  LDA #&T0     demander 'T0'
C86E  JSR $CFDB    C882  JSR $D067
C871  JSR $CE7A    C885  JSR $CF06    vérifier (enfin !) indice numérique
C874  JSR $CE77    C888  JSR $CF03    ramasser valeur limite
C877  LDA D5       C88B  LDA D5       prendre le signe
C879  ORA #7F      C88D  ORA #7F      ne pas toucher à b0-b6
C87B  AND D1       C88F  AND D1
C87D  STA D1       C891  STA D1       inclure le signe
C87F  LDA #8A      C893  LDA #9E      placer adresse de retour
C881  LDY #C8      C895  LDY #C8      soit #C88A/#C89E
C883  STA 91       C897  STA 91       en #91-#92
C885  STY 92       C899  STY 92
C887  JMP $CF34    C89B  JMP $CFC0    empiler ACC1 et retour ligne suivante

C88A  LDA #4B      C89E  LDA #81
C88C  LDY #DC      C8A0  LDY #DC      AY pointe sur la valeur 1 qui est le pas
C88E  JSR $DE73    C8A2  JSR $DE7B    pris par défaut: le placer dans ACC1
C891  JSR $00E8    C8A5  JSR $00E8    prendre caractère courant
C894  CMP #CB      C8A8  CMP #CB      est-ce STEP ?
C896  BNE C89E     C8AA  BNE C8B2     non,sauter
C898  JSR $00E2    C8AC  JSR $00E2    oui,le sauter
C89B  JSR $CE77    C8AF  JSR $CF03    évaluer une expression numérique
C89E  JSR $DF04    C8B2  JSR $DF13    prendre son signe dans A
C8A1  JSR $CF25    C8B5  JSR $CFB1    empiler A puis ACC1
C8A4  LDA B9       C8B8  LDA B9       sauver aussi l'adresse
C8A6  PHA          C8BA  PHA
```

```
C8A7  LDA B8      C8BB  LDA B8      de la variable d'indice
C8A9  PHA         C8BD  PHA
C8AA  LDA #8D     C8BE  LDA #8D     et enfin le code de FOR
C8AC  PHA         C8C0  PHA
```

EXECUTER UNE LIGNE

```
C8AD  JSR $C930   C8C1  JSR $C962   tester si Ctrl C
C8B0  LDA E9      C8C4  LDA E9
C8B2  LDY EA      C8C6  LDY EA      prendre TXTPTR dans AY
C8B4  BEQ C8BC    C8C8  BEQ C8D0    si mode direct,sauter
C8B6  STA AC      C8CA  STA AC      sinon,sauver pour CONT etc...
C8B8  STY AD      C8CC  STY AD      TXTPTR en #AC-D
C8BA  LDY #00     C8CE  LDY #00
C8BC  LDA (E9),Y  C8D0  LDA (E9),Y  prendre caractère courant
C8BE  BNE C918    C8D2  BNE C92F    si pas début de ligne sauter
..............    C8D4  LSR 0252    indiquer pas de IF pour l'instant
C8C0  LDY #02     C8D7  LDY #02
C8C2  LDA (E9),Y  C8D9  LDA (E9),Y  prendre Lien poids fort
C8C4  CLC         C8DB  CLC         préparer addition et pas d'erreur
C8C5  BNE C8CA    C8DC  BNE C8E1    si pas fin du programme,continuer
C8C7  JMP $C958   C8DE  JMP $C98A   sinon,traiter faire END
C8CA  INY         C8E1  INY
C8CB  LDA (E9),Y  C8E2  LDA (E9),Y  prendre numéro de la ligne courante
C8CD  STA A8      C8E4  STA A8      poids faible
C8CF  INY         C8E6  INY
C8D0  LDA (E9),Y  C8E7  LDA (E9),Y
C8D2  STA A9      C8E9  STA A9      et sauver aussi le poids fort
C8D4  TYA         C8EB  TYA
C8D5  ADC E9      C8EC  ADC E9      et ajuster TXTPTR
C8D7  STA E9      C8EE  STA E9      au début effectif de la ligne
C8D9  BCC C8DD    C8F0  BCC C8F4
C8DB  INC EA      C8F2  INC EA      sans oublier le pois fort
C8DD  BIT 02F4    C8F4  BIT 02F4    mode TRON ?
C8E0  BPL C8F5    C8F7  BPL C90C    non,sauter
C8E2  PHA         C8F9  PHA         sauver A (inutile)
C8E3  LDA #'('    C8FA  LDA #'('
C8E5  JSR $CC4D   C8FC  JSR $CCFB   afficher un crochet
C8E8  LDA A9      C8FF  LDA A9
C8EA  LDX A8      C901  LDX A8      puis le numéro de ligne
C8EC  JSR $E0C1   C903  JSR $E0C5   (#E0BD/#E0C1 serait plus judicieux)
C8EF  LDA #')'    C906  LDA #')'
C8F1  JSR $CC4D   C908  JSR $CCFB   fermer le crochet...
```

```
C8F4  PLA          C90B  PLA          récuperer A
C8F5  JSR $00E2    C90C  JSR $00E2    prendre caractère courant
C8F8  JSR $C8FE    C90F  JSR $C915    exécuter la commande ou l'affectation
C8FB  JMP $C8AD    C912  JMP $C8C1    et recommencer...
```

EXECUTER COMMANDE EN A

```
C8FE  BEQ C92D     C915  BEQ C960    si déjà fin de ligne,on sort
C900  SBC #80      C917  SBC #80     tester pour mot-clé
C902  BCC C915     C919  BCC C92C    si pas mot-clé,c'est une affectation
C904  CMP #42      C91B  CMP #42     #80+#42=#C2=dernière commande
C906  BCS C91C     C91D  BCS C94F    si pas commande, erreur
C908  ASL A        C91F  ASL A       calculer le déplacement
C909  TAY          C920  TAY         comme index
C90A  LDA C007,Y   C921  LDA C007,Y  prendre adresse de la commande poids fort
C90D  PHA          C924  PHA         et l'empiler
C90E  LDA C006,Y   C925  LDA C006,Y  idem pour le poids faible
C911  PHA          C928  PHA         l'empiler aussi
C912  JMP $00E2    C929  JMP $00E2   sauter la commande et exécution
C915  JMP $CAD2    C92C  JMP $CB1C   faire LET

C918  CMP #':'     C92F  CMP #':'    est-ce le séparateur ?
C91A  BEQ C8DD     C931  BEQ C8F4    oui,exécuter la commande suivante
.....              C933  CMP #C8     est-ce ELSE ?
.....              C935  BNE C945    non,sauter
.....              C937  BIT 0252    oui,y-a-t-il eu un IF ?
.....              C93A  BPL C94F    non,erreur
.....              C93C  JSR $CAB1   oui,trouver l'instruction suivante
.....              C93F  LSR 0252    et remettre drapeau IF
.....              C942  JMP $C8C1   continuer à l'instruction suivante
.....              C945  CMP #'''    est-ce REM abrégé ?
.....              C947  BNE C94F    non, erreur
.....              C949  JSR $CA99   oui,trouver la ligne suivante
.....              C94C  JMP $C8C1   et y continuer l'exécution
C91C  JMP $CFE4    C94F  JMP $D070   'SYNTAX ERROR'
```

'RESTORE' (COMMANDE)

Entrée: rien

Sortie: AY=#B0-#B1=Début basic-1

```
C91F  SEC        C952  SEC
C920  LDA 9A     C953  LDA 9A
C922  SBC #01    C955  SBC #01
C924  LDY 9B     C957  LDY 9B
C926  BCS C929   C959  BCS C95C    calculer dans AY
C928  DEY        C95B  DEY         le début du texte BASIC
C929  STA B0     C95C  STA B0
C92B  STY B1     C95E  STY B1      et le placer comme pointeur de DATA
C92D  RTS        C960  RTS
C92E  NOP        ..... ?
C92F  RTS        C961  RTS         ??
```

TESTER Ctrl C

Remarque: c'est la seule routine qui traite le Ctrl C.

```
C930  LDA 02DF   C962  LDA 02DF    prendre tampon touche
C933  BPL C92E   C965  BPL C960    si vide,on sort
C935  AND #7F    C967  AND #7F     sinon éliminer b7 (inutile)
C937  LDX #08    C969  LDX #08     inutile
C939  CMP #03    C96B  CMP #03     tester pour Ctrl C
C93B  BNE C92E   C96D  BNE C961    non,sortir (oui,C=1)
C93D  CMP #03    C96F  CMP #03     deux fois valent mieux qu'une ? Inutile
```

'STOP' (COMMANDE)

```
C93F  BCS C942   C971  BCS C974    garder C=1 (si pas de paramètre)
```

'END' (COMMANDE)

Bogue: sur la V1.1, l'imprimante n'est pas correctement mise hors service,ce qui peut causer des problèmes sur la longueur de ligne si on a mis l'imprimante en service à l'aide de la routine #C816.

```
C941  CLC        C973  CLC         Indiquer END,pas STOP
C942  BNE C987   C974  BNE C9B9    si des paramètres,sortir
C944  LDA E9     C976  LDA E9
C946  LDY EA     C978  LDY EA      prendre TXTPTR
C948  BEQ C956   C97A  BEQ C988    sauter si mode direct
C94A  STA AC     C97C  STA AC      et le sauver pour reprise éventuelle
C94C  STY AD     C97E  STY AD
```

```
C94E  LDA A8      C98Ø  LDA A8      sauver aussi le numéro de la ligne
C95Ø  LDY A9      C982  LDY A9
C952  STA AA      C984  STA AA
C954  STY AB      C986  STY AB      toujours pour reprise éventuelle
C956  PLA         C988  PLA         enlever l'adresse de retour
C957  PLA         C989  PLA         pour ajuster le pointeur de pile
C958  LDA #C1     C98A  LDA #BD
C95A  LDY #C3     C98C  LDY #C3     AY pointe sur 'BREAK'
C95C  LDX #ØØ     C98E  LDX #ØØ
C95E  STX Ø2F1    C99Ø  STX Ø2F1    indiquer imprimante OFF (mauvais pour V1.1)
C961  STX Ø2DF    C993  STX Ø2DF    vider tampon touche
C964  STX 2E      C996  STX 2E      pas de Ctrl O
C966  BCC C96B    C998  BCC C99D    sauter si END
C968  JMP $C4AA   C99A  JMP $C49D   si CTRL C ou STOP,afficher BREAK et retour
C96B  JMP $C4B5   C99D  JMP $C4A8   si END,saut direct à l'interpréteur.
```

'CONT' (COMMANDE)

Principe: si la reprise est autorisée, on place TXTPTR à son ancienne valeur (qui pointait sur un ØØ ou ':', sauf erreur de syntaxe. On ajuste aussi le numéro de ligne

```
C96E  BNE C987    C9AØ  BNE C9B9    sortir si il y a des paramètres
C97Ø  LDX #D7     C9A2  LDX #D7     préparer X pour 'CAN'T CONTINUE'
C972  LDY AD      C9A4  LDY AD      tester drapeau 'CONT'
C974  BNE C979    C9A6  BNE C9AB    sauter si <> Ø, c'est bon
C976  JMP $C485   C9A8  JMP $C47E   afficher l'erreur
C979  LDA AC      C9AB  LDA AC      prendre aussi le poids faible
C97B  STA E9      C9AD  STA E9
C97D  STY EA      C9AF  STY EA      et replacer TXTPTR
C97F  LDA AA      C9B1  LDA AA
C981  LDY AB      C9B3  LDY AB      prendre aussi numéro de ligne
C983  STA A8      C9B5  STA A8
C985  STY A9      C9B7  STY A9      et le réajuster aussi
C987  RTS         C9B9  RTS

C988  JMP $D2AØ   C9BA  JMP $D336   'ILLEGAL QUANTITY ERROR'
```

'RUN' (COMMANDE)

```
C98B  BNE C99Ø    C9BD  BNE C9C2    sauter s'il y a des paramètres
C98D  JMP $C733   C9BF  JMP $C7Ø8   sinon, placer TXTPTR et faire CLEAR
```

```
C99Ø  JSR $C73A   C9C2  JSR $C7ØF   faire CLEAR
C993  JMP $C9AA   C9C5  JMP $C9DC   puis GOTO
```

'GOSUB' (COMMANDE)

Principe: on sauve sur la pile TXTPTR et la ligne courante, qui permettent à eux seuls de retourner d'où on vient.

TXTPTR est sauvé avant l'évaluation du numéro de ligne qui suit le GOSUB, ceci pour se ramener au cas du GOTO. Ce qui veut dire que le RETURN devra sauter le numéro avant de relancer l'exécution.

L'adresse de retour à l'interpréteur reste sur la pile, contrairement au cas du FOR, ceci pour que le RETURN, lui, sorte par un simple RTS. Mais elle n'est plus au sommet, c'est pourquoi on saute directement à l'interpréteur.

```
C996  LDA #Ø3     C9C8  LDA #Ø3
C998  JSR $C43B   C9CA  JSR $C437   demander 6 octets sur la pile
C99B  LDA EA      C9CD  LDA EA      empiler TXTPTR poids fort
C99D  PHA         C9CF  PHA
C99E  LDA E9      C9DØ  LDA E9      puis poids faible
C9AØ  PHA         C9D2  PHA
C9A1  LDA A9      C9D3  LDA A9      empiler aussi le numéro de ligne poids fort
C9A3  PHA         C9D5  PHA
C9A4  LDA A8      C9D6  LDA A8      puis poids faible
C9A6  PHA         C9D8  PHA
C9A7  LDA #9B     C9D9  LDA #9B     et aussi token de GOSUB
C9A9  PHA         C9DB  PHA
C9AA  JSR $ØØE8   C9DC  JSR $ØØE8   prendre caractère courant
C9AD  JSR $C9B3   C9DF  JSR $C9E5   faire GOTO
C9BØ  JMP $C8AD   C9E2  JMP $C8C1   et retour à l'interpréteur
```

'GOTO' (COMMANDE)

Principe: la recherche du numéro de ligne commence á la ligne suivante si le numéro demandé est supérieur au numéro de la ligne courante, ceci pour accélérer la recherche.

De toutes faÇons, tout ce qui suit le numéro de ligne est ignoré et ne déclenche pas de SYNTAX ERROR

Bogue: pour savoir si la recherche doit démarrer à la ligne suivante, le test n'est fait que sur le poids fort ! Ce qui veut dire que pour le programme suivant, la recherche débutera au début du programme, parce que les poids forts des numéros de ligne sont égaux !

Exemple:200 GOTO 250: la recherche commence au début du programme, alors que pour la ligne:

200 GOTO 260, la recherche commence à la ligne suivante !

Curieux raccourci en tous cas !

```
C9B3  JSR $E79D   C9E5  JSR $E853   évaluer le numéro de ligne en #33-#34
C9B6  JSR $CA1F   C9E8  JSR $CA51   et chercher la fin de la ligne
C9B9  LDA A9      C9EB  LDA A9      prendre numéro courant poids fort
C9BB  CMP 34      C9ED  CMP 34      et comparer au numéro demandé
C9BD  BCS C9CA    C9EF  BCS C9FC   si inférieur ou égal, commencer au début
C9BF  TYA         C9F1  TYA         sinon, ajuster TXTPTR
C9C0  SEC         C9F2  SEC         sur la ligne suivante
C9C1  ADC E9      C9F3  ADC E9
C9C3  LDX EA      C9F5  LDX EA      pour pouvoir commencer la recherche
C9C5  BCC C9CE    C9F7  BCC CA00    à la ligne suivante et non au début
C9C7  INX         C9F9  INX         du programme
C9C8  BCS C9CE    C9FA  BCS CA00    inconditionnel
C9CA  LDA 9A      C9FC  LDA 9A      prendre début du programme comme
C9CC  LDX 9B      C9FE  LDX 9B      début de la recherche
C9CE  JSR $C6E8   CA00  JSR $C6BD   et rechercher l'adresse de la ligne
C9D1  BCC C9F1    CA03  BCC CA23    si pas trouvé, erreur
C9D3  LDA CE      CA05  LDA CE      ajuster TXTPTR
C9D5  SBC #01     CA07  SBC #01     sur le début de la ligne
C9D7  STA E9      CA09  STA E9
C9D9  LDA CF      CA0B  LDA CF      (#CE-#CF pointe après
C9DB  SBC #00     CA0D  SBC #00     le 0 du début de ligne)
C9DD  STA EA      CA0F  STA EA
C9DF  RTS         CA11  RTS
```

POP (COMMANDE)

Principe: la commande POP est traitée comme RETURN, la seule différence etant le positionnement de TXTPTR.

Lorsqu'on arrive sur une commande, Y contient le double du Token BASIC, ce qui permet de distinguer par la suite deux commandes ayant la même adresse. Ce procédé est aussi utilisé par les commandes graphiques et sonores, ainsi que PULL/UNTIL.

Après avoir réajusté la pile, on continue simplement l'exécution, sans toucher à TXTPTR.

RETURN (COMMANDE)

Principe: on recherche un bloc GOSUB sur la pile, et on récupère les valeurs de TXTPTR et du numéro de ligne. On retrouve alors l'adresse de retour à l'interpréteur qui avait été laissée lors du GOSUB.

Le GOSUB avait empilé TXTPTR avant l'évaluation du numéro de ligne. Ce qui explique que l'on calcule la fin de l'instruction via DATA

De même que GOTO, RETURN ne peut être suivi d'aucune instruction, ce qui explique que l'on recherche directement la fin de la ligne , via DATA.

La recherche du premier bloc GOSUB se fait en évitant les blocs FOR en simulant la recherche d'un bloc FOR qui aurait un indice dont l'adresse serait supérieure à #FFØØ, ce qui est impossible. Donc la recherche va s'arrêter sur le premier bloc qui n'est pas un bloc FOR.

```
C9EØ  BNE C9DF     CA12  BNE CA11     sortir s'il y a des paramètres.
C9E2  LDA #FF      CA14  LDA #FF      préciser une adresse
C9E4  STA B9       CA16  STA B9       qui est impossible
C9E6  JSR $C3CA    CA18  JSR $C3C6    et donc sauter tous les blocs FOR inutiles
C9E9  TXS          CA1B  TXS          et ajuster S au dessus
C9EA  CMP #9B      CA1C  CMP #9B      est-ce un bloc GOSUB ?
C9EC  BEQ C9F9     CA1E  BEQ CA2B     oui, ok
C9EE  LDX #16      CA2Ø  LDX #16      'RETURN WITHOUT GOSUB ERROR'
C9FØ  BYT $2C      CA22  BYT $2C      sauter instruction suivante
C9F1  LDX #$5A     CA23  LDX #$5A     'UNDEF'D STATEMENT ERROR'
C9F3  JMP $C485    CA25  JMP $C47E
C9F6  JMP $CFE4    CA28  JMP $DØ7Ø    'SYNTAX ERROR'
C9F9  PLA          CA2B  PLA          dépiler code GOSUB
C9FA  PLA          CA2C  PLA          et numéro de ligne poids faible
C9FB  CPY #8C      CA2D  CPY #8C      était-ce POP (#86%2=#1ØC --) #ØC)
C9FD  BEQ CA18     CA2F  BEQ CA4A     oui,sauter
C9FF  STA A8       CA31  STA A8       sauver numéro de ligne poids faible
CAØ1  PLA          CA33  PLA
CAØ2  STA A9       CA34  STA A9       idem poids fort
CAØ4  PLA          CA36  PLA          puis TXTPTR poids faible
CAØ5  STA E9       CA37  STA E9
CAØ7  PLA          CA39  PLA          et aussi poids fort
CAØ8  STA EA       CA3A  STA EA
```

'DATA' (COMMANDE)

Principe: l'instruction DATA ne fait rien, si ce n'est ralentir l'éxécution, puisqu'il faut la sauter simplement. C'est le READ qui permet de lire les DATA. C'est pourquoi il vaut mieux mettre les DATA à la fin d'un programme.

```
CA0A  JSR $CA1C    CA3C  JSR $CA4E   se placer sur l'instruction suivante
CA0D  TYA          CA3F  TYA         et ajuster
CA0E  CLC          CA40  CLC
CA0F  ADC E9       CA41  ADC E9      TXTPTR en conséquense
CA11  STA E9       CA43  STA E9
CA13  BCC CA17     CA45  BCC CA49
CA15  INC EA       CA47  INC EA
CA17  RTS          CA49  RTS         et retour à l'interpréteur
```

POP (SUITE)

```
CA18  PLA          CA4A  PLA         dépiler pour réajuster la pile
CA19  PLA          CA4B  PLA
CA1A  PLA          CA4C  PLA         et sortir simplement.
CA1B  RTS          CA4D  RTS
```

CALCUL DU DEPLACEMENT A LA PROCHAINE INSTRUCTION

```
CA1C  LDX #':'     CA4E  LDX #':'    second séparateur= ':'
CA1E  BYT $2C      CA50  BYT $2C     sauter instruction suivante
```

CALCUL DU DEPLACEMENT A LA FIN DE LA LIGNE

Entrée: le séparateur est dans X (00 sera toujours séparateur) et TXTPTR est correctement positionné.

Sortie: Y contient le déplacement au séparateur demandé.
Z=1
A contient le séparateur qui a arrêté la recherche.

Principe: selon le cas, il va falloir arrêter la recherche lorsqu'on rencontre un ':' ou non, c'est le deuxième séparateur.Le Null (00) est toujour séparateur.

Il ne faut pas prendre en compte les ':' compris à l'intérieur des chaines de caractéres. Ainsi le deuxième séparateur va-t-il être une alternativement ':' (extérieur d'une chaine) ou 00 (bidon car intercepté avant la comparaison) á l'intérieur d'une chaine.

```
CA1F  LDX #$00     CA51  LDX #$00     deuxième séparateur=00

CA21  STX 24       CA53  STX 24       sauver deuxième séparateur
CA23  LDY #00      CA55  LDY #00      préparer index
```

```
CA25  STY 25      CA57  STY 25      séparateur bascule=Ø au début
CA27  LDA 25      CA59  LDA 25
CA29  LDX 24      CA5B  LDX 24      inverser le deuxième séparateur
CA2B  STA 24      CA5D  STA 24
CA2D  STX 25      CA5F  STX 25
CA2F  LDA (E9),Y  CA61  LDA (E9),Y  prendre caractére
CA31  BEQ CA17    CA63  BEQ CA49    si nul (premier séparateur),on sort
CA33  CMP 25      CA65  CMP 25      si deuxième séparateur
CA35  BEQ CA17    CA67  BEQ CA49    on sort aussi
CA37  INY         CA69  INY         préparer pour caractère suivant
CA38  CMP #'''    CA6A  CMP #'''    et tester si début ou fin de chaine
CA3A  BNE CA2F    CA6C  BNE CA61    non, on continue
CA3C  BEQ CA27    CA6E  BEQ CA59    oui, on inverse les séparateurs
```

'IF' (COMMANDE)

Principe: IF appelle l'évaluation normale sans se préoccuper de son type. Les opérateurs de relation (=,),(ne sont donc pas obligatoires. Le test est ensuite fait sur #DØ, qui contient l'exposant pour une nombre et la longueur de la chaine si ... c'est une chaine.

La valeur est considérée comme vraie (exécution du THEN ou du GOTO) si #DØ<>Ø c'est à dire chaine non vide ou nombre non nul

La valeur est considérée comme fausse (exécution du ELSE ou directement de la ligne suivante) si #DØ=Ø c'est à dire si le nombre est nul ou si la chaine est vide

Remarque: s'il est vrai que toute valeur non nulle est considérée comme vraie, il faut faire attention à l'usage de l'opérateur NOT: NOT 2=-3, est vrai car non nul aussi...

```
CA3E  JSR $CE8B   CA7Ø  JSR $CF17   évaluer la condition
CA41  JSR $ØØE8   CA73  JSR $ØØE8   prendre caractére suivant la condition
CA44  CMP #97     CA76  CMP #97     est-ce GOTO ?
CA46  BEQ CA4D    CA78  BEQ CA7F    oui, sauter
CA48  LDA #C9     CA7A  LDA #C9     non, on veut un THEN
CA4A  JSR $CFDB   CA7C  JSR $DØ67
CA4D  LDA DØ      CA7F  LDA DØ      prendre résultat de la condition
CA4F  BNE CA56    CA81  BNE CA88    condition vraie, sauter
CA51  JSR $CA66   CA83  JSR $CA9E   chercher un ELSE
CA54  BEQ CAØD    CA86  BEQ CA3F    si pas trouvé, aller ligne suivante
CA56  JSR $ØØE8   CA88  JSR $ØØE8   prendre carctére courant
CA59  BCS CA5E    CA8B  BCS CA9Ø    si ce n'est pas un chiffre,sauter
CA5B  JMP $C9B3   CA8D  JMP $C9E5   sinon, faire GOTO
```

```
..... CA9Ø  PHP        sauver le type du caractère
..... CA91  SEC        inutile: on vient d'un BCS !
..... CA92  ROR Ø252   mettre drapeau IF pour ignorer les ELSE
..... CA95  PLP        récuperer type du caractère
CA5E JMP $C8FE  CA96  JMP $C915  exécuter commande en A
```

REM (COMMANDE)

Principe: l'instruction REM ne fait pas grand chose, à l'instar de DATA. Le programme continuant son exécution à la ligne suivante, on comprend qu'il soit inutile de faire suivre une REM d'une instruction quelconque...

```
CA61 JSR $CA1F   CA99  JSR $CA51  calculer déplacement à la ligne suivante
CA64 BEQ CAØD    CA9C  BEQ CA3F   inconditionnel: ajuster TXTPTR et sortir
```

IF (suite): TROUVER le ELSE

```
CA66 LDY #ØØ     CA9E  LDY #ØØ     préparer index
CA68 LDA (E9),Y  CAAØ  LDA (E9),Y  prendre le caractére
CA6A BEQ CA74    CAA2  BEQ CABØ    si fin de ligne, on sort
CA6C INY         CAA4  INY         préparer pour caractère suivant
..... CAA5  CMP #C7     est-ce THEN ?
..... CAA7  BEQ CA99    oui, se placer à la fin de la ligne
CA6D CMP #C8     CAA9  CMP #C8     est-ce ELSE ?
CA6F BNE CA68    CAAB  BNE CAAØ    non, on continue la recherche
CA71 JMP $CAØD   CAAD  JMP $CA3F   oui, on se positionne juste après
CA74 RTS         CABØ  RTS
```

??????????????????

Action: trouve la fin d'une instruction, mais les ':' entre guillemets sont aussi considérés comme des séparateurs.

Routine jamais appelée: Curieux ajout...

```
..... CAB1  LDY #FF     préparer l'index
..... CAB3  INY         passer caractère suivant
..... CAB4  LDA (E9),Y  prendre caractère
..... CAB6  BEQ CABC    si fin de ligne, on se place dessus
..... CAB8  CMP #':'    est-ce un séparateur ?
..... CABA  BNE CAB3    non, on continue
..... CABC  JMP $CA3F   oui, placer TXTPTR à la fin

CA75 JMP $CFE4   CABF  JMP $DØ7Ø  'SYNTAX ERROR'
```

Principe: après avoir évalué le déplacement, on sauve le code GOSUB ou GOTO puis on commence la recherche, en sautant de virgule en virgule.

Si le déplacement est trop long,on se retrouve à la fin de l'instruction et on sort sans rien faire.

Si il est nul, le premier DEC #D4 le ramène à #FF, et on se retrouve donc dans le cas d'un déplacement trop long.

Astucieuse aussi cette manière de simuler un GOTO ou GOSUB simple en repartant directement à l'interpréteur.

Tous les numéros de ligne sont évalués jusqu'à ce que le bon soit trouvé, ce qui implique une certaine lenteur si on doit atteindre le dernier...

```
CA78  JSR $D80D   CAC2  JSR $D8C8   évaluer en #D4 le déplacement
CA7B  PHA         CAC5  PHA         sauver le code (GOSUB ou GOTO)
CA7C  CMP #9B     CAC6  CMP #9B     GOSUB ?
CA7E  BEQ CA84    CAC8  BEQ CACE    oui,sauter
CA80  CMP #97     CACA  CMP #97     GOTO ?
CA82  BNE CA75    CACC  BNE CABF    non, erreur
CA84  DEC D4      CACE  DEC D4      prendre numéro suivant
CA86  BNE CA8C    CAD0  BNE CAD6    si pas le bon, continuer
CA88  PLA         CAD2  PLA         récupérer le code d'instruction
CA89  JMP $C900   CAD3  JMP $C917   et exécuter la commande

CA8C  JSR $00E2   CAD6  JSR $00E2   sauter ',' ou GOTO/GOSUB (1ère fois)
CA8F  JSR $CA98   CAD9  JSR $CAE2   ramasser le numéro de ligne
CA92  CMP #','    CADC  CMP #','    si il est suivi d'une virgule
CA94  BEQ CA84    CADE  BEQ CACE    on continue la recherche
CA96  PLA         CAE0  PLA         réajuster la pile si déplacement trop grand
CA97  RTS         CAE1  RTS         et sortir
```

EVALUER UN NUMERO DE LIGNE EN #33-#34

Entrée: C et A doivent être positionnés comme au sortir d'un JSR #00E2 ou #00E8: 0 si chiffre, 1 sinon.

Sortie: #33-#34 contient la valeur, une erreur étant générée si le numéro dépasse 64000.

Y n'est pas touché par cette routine

TXTPTR pointe sur le premier caractère non chiffre, et la sortie se fait par un JSR #00E2 (A,N et Z bien placés).

Principe: au fur et à mesure des caractères, on multiplie par 10 le résultat,

de sorte que le nouveau caractère se retrouve bien comme chiffre des unités.

Le test de dépassement est fait avant multiplication. #1A00=6400=64000/10.6 400 est pratique car il suffit de tester le poids fort pour savoir s'il y a dépassement.64000=#FA00 est aussi un nombre 'rond' mais il aurait fallu rajouter un LDA $34 pour tester le poids fort après le produit par 10, c'est pour ceci que le test est fait avant multiplication.

Bref,beaucoup de chance que 64000 et 6400 soient des nombres 'ronds' aussi bien en Hexa qu'en décimal...

```
CA98  LDX #00      CAE2  LDX #00      mettre à 0 le résultat...
CA9A  STX 33       CAE4  STX 33
CA9C  STX 34       CAE6  STX 34
CA9E  BCS CA97     CAE8  BCS CAE1    sortir si caractère non numérique
CAA0  SBC #2F      CAEA  SBC #2F     ramener à 0-9 (C=0 donc soustrait #30)
CAA2  STA 24       CAEC  STA 24      et sauver
CAA4  LDA 34       CAEE  LDA 34      poids fort du résultat
CAA6  STA 91       CAF0  STA 91      et le sauver
CAA8  CMP #19      CAF2  CMP #19     si résultat )= #1900=6400, plus de chiffre
CAAA  BCS CA80     CAF4  BCS CACA    alors 'SYNTAX ERROR' (A ne peut valoir #97)
CAAC  LDA 33       CAF6  LDA 33      poids faible résultat
CAAE  ASL A        CAF8  ASL A       x2
CAAF  ROL 91       CAF9  ROL 91      poids fort aussi
CAB1  ASL A        CAFB  ASL A       x4
CAB2  ROL 91       CAFC  ROL 91      poids fort aussi
CAB4  ADC 33       CAFE  ADC 33      +original:x5
CAB6  STA 33       CB00  STA 33
CAB8  LDA 91       CB02  LDA 91      récupérer le poids fort
CABA  ADC 34       CB04  ADC 34      ajouter original et report aussi
CABC  STA 34       CB06  STA 34      x5 ,donc
CABE  ASL 33       CB08  ASL 33      x2x5=x10
CAC0  ROL 34       CB0A  ROL 34      et on n'oublie pas le poids fort
CAC2  LDA 33       CB0C  LDA 33      on ajoute le nouveau venu
CAC4  ADC 24       CB0E  ADC 24      qui prend la place des unités
CAC6  STA 33       CB10  STA 33
CAC8  BCC CACC     CB12  BCC CB16
CACA  INC 34       CB14  INC 34      on n'oublie pas non plus le poids fort
CACC  JSR $00E2    CB16  JSR $00E2   on prend le caracère suivant
CACF  JMP $CA9E    CB19  JMP $CAE8   et on recommence
```

'LET' (COMMANDE)

Principe: l'adresse où doivent être stockées les données est rapidement mise en #B8-#B9. Le transfert pour les variables numérique est trivial, il n'en est pas

de même pour le transfert des chaines...

Plusieurs cas sont à distinguer:

1)type A$=B$: la chaine est déjà dans la zone des chaines, et le descripteur aussi, il n'y a pas eu de réservation temporaire. Celle-ci va donc être faite, et on y transfère la chaine.

2)type A$="ORIC". Dans ce cas le descripteur est dans la pile. Il faut donc l'enlever de la pile mais il est inutile de réserver de la place puisque l'adresse de la chaine est directement dans le programme Basic.

3)type A$="ORIC"+"-1". Dans ce cas il y a une réservation temporaire, il faut donc enlever le descripteur de la pile mais il est inutile de réserver de la place car celà a déjà été fait par la routine d'évaluation.

Dans tous les cas, la réservation est enlevée, si elle existait, et le descripteur est transféré dans la variable.

NB:en mode direct, le type 2 est ramené au type 3 lors de l'évaluation de la chaine.

```
CAD2  JSR $D0FC    CB1C  JSR $D188   Prendre la variable
CAD5  STA B8       CB1F  STA B8
CAD7  STY B9       CB21  STY B9      et sauver son adresse
CAD9  LDA #&=      CB23  LDA #&=     demander un '='
CADB  JSR $CFDB    CB25  JSR $D067   (codé évidemment)
CADE  LDA 29       CB28  LDA 29      sauver type première variable
CAE0  PHA          CB2A  PHA
CAE1  LDA 28       CB2B  LDA 28      idem pour le drapeau chaine/numérique
CAE3  PHA          CB2D  PHA
CAE4  JSR $CE8B    CB2E  JSR $CF17   évaluer la valeur à affecter
CAE7  PLA          CB31  PLA         récupérer drapeau
CAE8  ROL A        CB32  ROL A       et le mettre dans C
CAE9  JSR $CE7D    CB33  JSR $CF09   vérifier même type
CAEC  BNE CB06     CB36  BNE CB50    sauter si chaine
CAEE  PLA          CB38  PLA         récupérer drapeau entier/réel
CAEF  BPL CB03     CB39  BPL CB4D    et sauter si réel
CAF1  JSR $DEEC    CB3B  JSR $DEF4   arrondir ACC1
CAF4  JSR $D217    CB3E  JSR $D2A9   et convertir en entier signé dans #D4-#D3
CAF7  LDY #00      CB41  LDY #00     préparer l'index
CAF9  LDA D3       CB43  LDA D3      prendre poids fort
CAFB  STA (B8),Y   CB45  STA (B8),Y  et le mettre dans la variable
CAFD  INY          CB47  INY
CAFE  LDA D4       CB48  LDA D4      puis prendre poids faible
CB00  STA (B8),Y   CB4A  STA (B8),Y  et le sauver aussi
CB02  RTS          CB4C  RTS         et c'est tout !

CB03  JMP $DEA1    CB4D  JMP $DEA9   AACC! -- > (#B8-#B9)
```

```
Traiter les chaines

CB06  PLA          CB50  PLA          ajuster la pile
CB07  LDY #02      CB51  LDY #02      indexer poids fort adresse chaine
CB09  LDA (D3),Y   CB53  LDA (D3),Y   et le prendre
CB0B  CMP A3       CB55  CMP A3       la chaine est-elle
CB0D  BCC CB26     CB57  BCC CB70     stockée dans la zone des chaines ?
CB0F  BNE CB18     CB59  BNE CB62     oui, sauter
CB11  DEY          CB5B  DEY
CB12  LDA (D3),Y   CB5C  LDA (D3),Y
CB14  CMP A2       CB5E  CMP A2       comparer aussi les poids faibles...
CB16  BCC CB26     CB60  BCC CB70     si chaine hors de la zone, sauter

Traiter chaine déjà réservée

CB18  LDY D4       CB62  LDY D4       faut-il faire une réservation, soit
CB1A  CPY 9D       CB64  CPY 9D       le descripteur est-il déjà dans la zone
CB1C  BCC CB26     CB66  BCC CB70     des variables ?
CB1E  BNE CB2D     CB68  BNE CB77     oui, sauter
CB20  LDA D3       CB6A  LDA D3
CB22  CMP 9C       CB6C  CMP 9C       comparer aussi les poids faibles
CB24  BCS CB2D     CB6E  BCS CB77
CB26  LDA D3       CB70  LDA D3       oui, reprendre adresse du descripteur
CB28  LDY D4       CB72  LDY D4       pour préparer le transfert des descripteurs
CB2A  JMP $CB43    CB74  JMP $CB8D    et finir (BCC aurait été le bienvenu)

Réserver la chaine et transfert

CB2D  LDY #00      CB77  LDY #00
CB2F  LDA (D3),Y   CB79  LDA (D3),Y   prendre longueur de la chaine
CB31  JSR $D4E8    CB7B  JSR $D5A3    et réserver la place
CB34  LDA BF       CB7E  LDA BF       récupérer l'adresse du descripteur
CB36  LDY C0       CB80  LDY C0       laissée par la routine de réservation
CB38  STA DE       CB82  STA DE       et préparer le transfert (la routine de
CB3A  STY DF       CB84  STY DF       réservation a laissé l'adresse en #A4-#A5)
CB3C  JSR $D6E9    CB86  JSR $D7A4    et transférer la chaine vers zone réservée
CB3F  LDA #D0      CB89  LDA #D0      prendre l'adresse du descripteur
CB41  LDY #00      CB8B  LDY #00      à éliminer

Enlever descripteur temporaire et le placer dans la variable

CB43  STA BF       CB8D  STA BF       sauver l'adresse du descripteur pour
CB45  STY C0       CB8F  STY C0       transfert prochain et irrémédiable...
CB47  JSR $D74A    CB91  JSR $D805    dépiler éventuel descripteur temporaire
```

```
CB4A  LDY #ØØ     CB94  LDY #ØØ     et faire le transfert du descripteur
CB4C  LDA (BF),Y  CB96  LDA (BF),Y  longueur de la chaine
CB4E  STA (B8),Y  CB98  STA (B8),Y  vers variable
CB5Ø  INY         CB9A  INY
CB51  LDA (BF),Y  CB9B  LDA (BF),Y  adresse poids faible
CB53  STA (B8),Y  CB9D  STA (B8),Y
CB55  INY         CB9F  INY
CB56  LDA (BF),Y  CBAØ  LDA (BF),Y  et poids fort aussi...
CB58  STA (B8),Y  CBA2  STA (B8),Y
CB5A  RTS         CBA4  RTS         et finir
CB5B  JSR $CBFØ   CBA5  JSR $CCB3   afficher la chaine évaluée
CB5E  JSR $ØØE8   CBA8  JSR $ØØE8   et prendre caractère courant pour continuer
```

'PRINT' (COMMANDE)

Remarque: c'est une des routines qui a le plus changé de la V1.Ø vers la V1.1, la plupart des bogues ont disparues, notament ce décalage de 13 colonnes de la tabulation et le fait qu'il était impossible d'écrire sur la dernière colonne.

Principe: les expressions sont évaluées, et affichées directement si ce sont des chaines ou converties en chaines puis affichées si ce sont des nombres. Si des caractères 'de controle' (, ; TAB(SPC( ) apparaissent, ils sont traités à part. Notons que le ';' est le plus souvent inutile puisqu'il ne fait rien, sauf séparer les variables...

```
CB61  BEQ CB9F    CBAB  BEQ CBFØ   pas (ou plus) de paramètre,aller à la ligne
CB63  BEQ CBAB    CBAD  BEQ CCØB   si fin PRINT, sortir sans aller à la ligne
CB65  CMP #&TAB(  CBAF  CMP #&TAB(  est-ce TAB( ?
CB67  BEQ CBCA    CBB1  BEQ CC2E    oui, sauter,C=1
CB69  CMP #&SPC(  CBB3  CMP #&SPC(  est-ce SPC( ?
CB6B  CLC         CBB5  CLC
CB6C  BEQ CBCA    CBB6  BEQ CC2E    oui, sauter, C=Ø
CB6E  CMP #','    CBB8  CMP #','    est-ce la virgule ?
CB7Ø  BEQ CBAC    CBBA  BEQ CCØC    oui, sauter
CB72  CMP #';'    CBBC  CMP #';'    est-ce ';' ?
CB74  BEQ CBDF    CBBE  BEQ CC2B    oui, sauter
.............     CBCØ  CMP #&'@'   est-ce '@' ?
.............     CBC2  BNE CBC7    non, continuer
.............     CBC4  JMP $CC59   oui, sauter vers @
CB76  JSR $CE8B   CBC7  JSR $CF17   évaluer l'expression
CB79  BIT 28      CBCA  BIT 28      est-ce une chaine ?
CB7B  BMI CB5B    CBCC  BMI CBA5    oui, afficher une chaine
CB7D  JSR $EØD1   CBCE  JSR $EØD5   convertir ACC1 en chaine (AY vaut #1ØØ)
CB8Ø  JSR $D4FA   CBD1  JSR $D5B5   évaluer la chaine pointée par AY
```

```
CB83  LDY #00     CBD4  LDY #00     le nombre tient-il sur la ligne ?
CB85  LDA (D3),Y  CBD6  LDA (D3),Y  prendre longueur
CB87  CLC         CBD8  CLC
CB88  ADC 30      CBD9  ADC 30      et l'ajouter à la position du curseur
CB8A  CMP 31      CBDB  CMP 31      et comparer à maxi permis
CB8C  BCC CB91    CBDD  BCC CBE2    c'est bon, sauter
CB8E  JSR $CB9F   CBDF  JSR $CBF0   trop long, on va à la ligne
CB91  JSR $CBF0   CBE2  JSR $CCB3   et on affiche la chaine
CB94  JSR $CC0D   CBE5  JSR $CCD4   suivie d'un espace
CB97  BNE CB5E    CBE8  BNE CBA8    inconditionnel: et on continue...
```

TERMINER UN ORDRE DANS LE TAMPON CLAVIER

Cette routine a été placée là pour une raison inconue, il faut bien le dire.

```
CB99  LDY #00     CBEA  LDY #00
CB9B  STY 35,X    CBEC  STY 35,X    mettre terminateur 00 à la fin de l'ordre
CB9D  LDX #34     CBEE  LDX #34     préparer X pour début du tampon (XY=#0034)
```

ALLER A LA LIGNE

Entrée: rien de spécial

Sortie: X et Y n'ont pas été touchés.

Principe: on envoie un retour chariot, et on réinitialise (on essaie !) la position courante à 0. C'est la seule routine qui le fasse: si un Retour chariot (#0D) est rencontré pendant l'affichage d'une chaine sur l'imprimante, l'octet #30 n'est pas remis à jour (du moins pour l'imprimante), ce qui peut être génant.

Bogue: la position courante est réinitialisée à #0D (13) au lieu de 00 sur la V1.0. Voilá pourquoi le TAB(est décalé de 13 octets !

```
CB9F  LDA #0D     ..............  code RC
CBA1  STA 30      ..............  n'importe quoi !
CBA3  JSR $CC12   ..............  et envoyer le RC
CBA6  LDA #0A     ..............  saut de ligne
CBA8  JSR $CC12   ..............  et envoyer le saut de ligne
CBAB  RTS         ..............  JMP $CC12 aurait fait aussi bien !
```

Le traitement différent appliqué à l'imprimante s'explique par le fait qu'à

l'écran, le curseur précède toujours le caractère à afficher, alors que l'imprimante, elle, ne gère pas de curseur.

```
..... CBF0 LDA 30     prendre la position courante du curseur
..... CBF2 PHA        et la sauver
..... CBF3 LDA #0D    en tous les cas,
..... CBF5 JSR $CCD9  envoyer un retour chariot
..... CBF8 PLA        récupérer position
..... CBF9 BIT 02F1   sommes-nous sur l'imprimante ?
..... CBFC BMI CC02   oui, on saute
..... CBFE CMP 31     non, étions-nous sur la dernière colonne ?
..... CC00 BEQ CC0B   alors c'est tout pour aujourd'hui
..... CC02 LDA #00    réinitialiser
..... CC04 STA 30     la position courante
..... CC06 LDA #0A    et envoyer un saut de ligne
..... CC08 JSR $CCD9
..... CC0B RTS
CBAC NOP        ..... traiter la ','
CBAD NOP        .....
CBAE NOP        ..... pas très intéressant tout ça !
CBAF NOP        .....
CBB0 NOP        .....
CBB1 NOP        .....
CBB2 NOP        .....
CBB3 LDA 30     ..... prendre position courante
CBB5 CMP 32     ..... et comparer à maxi tabulation
CBB7 BCC CBBF   ..... si pas atteint, on continue
CBB9 JSR $CB9F  ..... sinon, on va à la ligne
CBBC JMP $CBBF  ..... et on continue après la virgule
CBBF SEC        ..... calculer la distance à une colonne
CBC0 SBC #08    ..... multiple de 8
CBC2 BCS CBC0   ..... continuer tant que pas #F8 à #FF
CBC4 EOR #FF    ..... complémenter: #00 à #07
CBC6 ADC #01    ..... et enfin #01 à #08
CBC8 BNE CBDA   ..... inconditionnel: afficher A espaces
..... CC0C LDA 30     traiter la , : prendre position courante
..... CC0E BIT 02F1   est-ce sur l'imprimante ?
..... CC11 BMI CC17   oui, sauter
..... CC13 SEC        écran
..... CC14 SBC 0253   enlever éventuellement la protection des
..... CC17 SEC        colonnes de droite
..... CC18 SBC #08    et calculer la distance à une colonne
..... CC1A BCS CC18   multiple de 8
..... CC1C EOR #FF    #00 à #07
```

```
..... CC1E ADC #01     #01 à #08
..... CC20 TAX         et nombre d'espaces à afficher dans X
..... CC21 CLC
..... CC22 ADC 30      plus position courante
..... CC24 CMP 31      dépasse-t-on le maxi permis ?
..... CC26 BCC CC47    non, c'est bon
..... CC28 JSR $CBF0   oui, aller à la ligne
..... CC2B JMP $CC4B   sauter la virgule et continuer
```

Traiter TAB(et SPC(

```
CBCA PHP       CC2E PHP       traiter TAB( (C=1) et SPC( (C=0)
CBCB JSR $D80A CC2F JSR $D8C5 évaluer l'argument
CBCE CMP #')'  CC32 CMP #')'  a-t-on bien fermé la parenthèse ?
CBD0 BNE CBEA  CC34 BNE CC56  non, erreur
CBD2 PLP       CC36 PLP       au fait, TAB( ou SPC( ?
CBD3 BCC CBDB  CC37 BCC CC47  sauter si SPC(
CBD5 TXA       CC39 TXA       et prendre dans A l'argument
.....          CC3A CMP 31    est-il plus grand que le maxi ?
.....          CC3C BCC CC41  non, c'est bon
.....          CC3E JMP $D336 oui,'ILLEGAL QUANTITY ERROR'
.....          CC41 SEC       réajuster C
CBD6 SBC 30    CC42 SBC 30    enlever la position courante
CBD8 BCC CBDF  CC44 BCC CC4B  si en dessous, il ne se passe rien
CBDA TAX       CC46 TAX       sinon, nombre d'espaces à afficher dans X
CBDB INX       CC47 INX       et ajuster
CBDC DEX       CC48 DEX       espace suivant
CBDD BNE CBE5  CC49 BNE CC51  si pas fini, continuer
CBDF JSR $00E2 CC4B JSR $00E2 sauter la , ou ; ou )
CBE2 JMP $CB63 CC4E JMP $CBAD et continuer, sauf si fin du PRINT
CBE5 JSR $CC0D CC51 JSR $CCD4 afficher un espace
CBE8 BNE CBDC  CC54 BNE CC48  inconditionnel: et continuer
CBEA JMP $CFE4 CC56 JMP $D070 afficher 'SYNTAX ERROR'
```

Traiter @

Bogue: la position de tabulation (#30) n'est pas initialisée,et donc les tabulations s'avèreront hasardeuses après un PRINT @

```
..... CC59 BIT 02F1    traiter @
..... CC5C BMI CC56    si mode imprimante, erreur
..... CC5E LDX 021F    si mode HIRES
..... CC61 BEQ CC66
..... CC63 JMP $EAF7   'DISP TYPE MISMATCH'
```

```
.....  CC66  JSR $D8C5   évaluer coordonnée horizontale
.....  CC69  CPX #28
.....  CC6B  BCS CCAD    si ) 40, ILLEGAL QUANTITY (chemin détourné)
.....  CC6D  STX 0C      et stocker temporairement
.....  CC6F  JSR $D065   demander une ','
.....  CC72  JSR $D8C8   et évaluer la coordonnée verticale
.....  CC75  INX         et ajuster car 0=ligne 1 (!)
.....  CC76  CPX #1C     ligne )28 ?
.....  CC78  BCS CCAD    oui,erreur
.....  CC7A  LDA 026A    sauver
.....  CC7D  PHA         la couleur du curseur
.....  CC7E  AND #FE     et éteindre le curseur
.....  CC80  STA 026A    pour éviter les 'traces'
.....  CC83  LDA #00     ne pas oublier de l'effacer
.....  CC85  JSR $F801
.....  CC88  LDA 0C      récupérer coordonnée horizontale
.....  CC8A  STA 0269    nouvelle coordonnée curseur
.....  CC8D  TXA
.....  CC8E  STA 0268    idem pour coordonnée horizontale
.....  CC91  JSR $DA0C   calculer l'adresse de la ligne
.....  CC94  LDA 1F      inutile (le poids faible est déjà dans A)
.....  CC96  LDY 20
.....  CC98  STA 12
.....  CC9A  STY 13      et la placer comme nouvelle adresse
.....  CC9C  PLA         récupérer
.....  CC9D  STA 026A    la couleur originale du curseur
.....  CCA0  LDA #01
.....  CCA2  JSR $F801   et éventuellement l'afficher
.....  CCA5  LDA #';'    demander un ';'
.....  CCA7  JSR $D067
.....  CCAA  JMP $CBAD   et continuer le PRINT
.....  CCAD  JMP $D8C2   'ILLEGAL QUANTITY ERROR'
```

IMPRIMER UNE CHAINE POINTEE PAR AY

Entrée: AY pointe sur une chaine (terminée par un #00).

Sortie: après évaluation par la routine standards, la chaine est affichée sur l'écran ou l'imprimante selon #2F1.

```
CBED  JSR $D4FA   CCB0  JSR $D5B5   évaluer la chaine pointée par AY
```

IMPRIMER UNE CHAINE DONT LE DESCRIPTEUR EST DANS ACC1

Bogue: lorsqu'on affiche un caractère à l'écran, l'octet #3Ø est toujours à jour. En revanche cette remise à jour n'est pas faite pour l'imprimante, de sorte que si on envoie des RC, il y aura des problèmes de tabulation. Le RC a tout de même failli être bien géré !

```
CBFØ  JSR $D715    CCB3  JSR $D7DØ    prendre adresse et enlever réservation
CBF3  TAX          CCB6  TAX          longueur dans X
CBF4  LDY #ØØ      CCB7  LDY #ØØ      préparer index
CBF6  INX          CCB9  INX          préparer la suite...
CBF7  DEX          CCBA  DEX          compter les caractères
CBF8  BEQ CBAB     CCBB  BEQ CCCD     si fin, on sort
CBFA  LDA (91),Y   CCBD  LDA (91),Y   prendre caractère
CBFC  JSR $CC12    CCBF  JSR $CCD9    et l'afficher
CBFF  INY          CCC2  INY          préparer caractère suivant
CCØØ  CMP #ØD      CCC3  CMP #ØD      est-ce un retour chariot ?
CCØ2  BNE CBF7     CCC5  BNE CCBA     non, on continue
CCØ4  JSR $CBAB    CCC7  JSR $CC0B    =RTS ! très utile.
CCØ7  JMP $CBF7    CCCA  JMP $CCBA    et continuer
.....              CCCD  RTS          il y en avait un pas si loin, en #CD12
```

o 'CLS' (COMMANDE)

```
CCØA  LDA #ØC      CCCE  LDA #ØC      prendre code pour Ctrl L
CCØC  BYT #2C      CCDØ  BYT #2C      et sauter les instructions suivantes
```

INVERSER CLIGNOTEMENT DU CURSEUR

```
.....              CCD1  LDA #11      prendre code pour Ctrl Q
.....              CCD3  BYT #2C      et sauter les instructions suivantes
```

AFFICHER UN ESPACE

```
CCØD  LDA #' '     CCD4  LDA #' '     prendre code pour espace
CCØF  BYT #2C      CCD7  BYT #2C      et sauter l'instruction suivante
```

AFFICHER UN '?'

```
CC1Ø  LDA #'?'     CCD7  LDA #'?'     prendre code pour '?'
```

AFFICHER LE CARACTERE DANS A

Entrée: A contient le caractère à afficher. Les différences avec la routine du moniteur sont: d'une part cette dernière ne gère pas l'imprimante, et d'autre part elle ne gère pas non plus la tabulation (#3Ø).

Sortie: le caractère est affiché sur l'imprimante ou l'écran selon #2F1.
A,X,Y sont inchangés.
Z et N sont positionnés selon A.
Attention: #27 sert pour la V1.Ø à sauver A et pour la V1.1 à sauver X.

Remarque: la V1.Ø retourne un message (qui n'interrompt pas le programme) si l'imprimante n'a pas acquitté le caractère après 15 secondes, ce qui est pratique et permet de sortir d'un LPRINT malencontreux.

```
CC12  BIT 2E      CCD9  BIT 2E      Ctrl O actif ?
CC14  BMI CC6A    CCDB  BMI CD1Ø    oui, on sort tout de suite
CC16  BIT Ø2F1    ..............    est-on sur l'imprimante ?
CC19  BPL CC4D    ..............    non, sauter
CC1B  PHA         CCDD  PHA         oui, sauver le code à imprimer
CC1C  CMP #2Ø     CCDE  CMP #2Ø     est-ce un caractère de controle ?
CC1E  BCC CC2B    CCEØ  BCC CCED    oui, sauter la tabulation
CC2Ø  LDA 3Ø      CCE2  LDA 3Ø      prendre position tabulation
CC22  CMP 31      CCE4  CMP 31      maximum atteint ?
CC24  BNE CC29    CCE6  BNE CCEB    non, c'est bon
CC26  JSR $CB9F   CCE8  JSR $CBFØ   oui, aller à la ligne
CC29  INC 3Ø      CCEB  INC 3Ø      on incrémente la tabulation
CC2B  PLA         CCED  PLA         avant de récupérer le code pour l'afficher
CC2C  STA 27      ..............    sauver le caractère à afficher
CC2E  TXA         ..............    sauver X...
CC2F  PHA         ..............
CC3Ø  TYA         ..............    sauver Y...
CC31  PHA         ..............
CC32  LDA 27      ..............    récupérer le code
CC34  JSR $F433   ..............    et l'envoyer à l'imprimante
CC37  TAX         ..............    attente de l'ACK trop longue peut être ?
CC38  BPL CC44    ..............    non, c'est bon (TAX inutile)
CC3A  LDA #6D     ..............
CC3C  LDY #CC     ..............    AY pointe sur '?PRINTER ERROR'
CC3E  LSR Ø2F1    ..............    on met l'imprimante hors service
CC41  JSR $CBED   ..............    et on affiche le message
CC44  PLA         ..............
CC45  TAY         ..............    récupérer Y
CC46  PLA         ..............
```

```
CC47  TAX          ..............  et X
CC48  LDA 27       ..............  et A
CC4A  AND #FF      ..............  positionner N et Z
CC4C  RTS          ..............
CC4D  STA 27       ..............  sauver le caractère
CC4F  TYA          ..............
CC5Ø  PHA          ..............  et Y
CC51  TXA          ..............
CC52  PHA          ..............  et X
CC53  LDA 27       ..............
CC55  CMP #2Ø      ..............  caractère de controle ?
CC57  BCC CC6Ø     ..............  oui, on saute
CC59  TAY          ..............
CC5A  INY          ..............  est-ce DEL (#7F+1=#8Ø)
CC5B  BMI CC6Ø     ..............  oui, sauter
CC5D  ORA Ø2F7     ..............  faire un ORA ????!
CC6Ø  TAX          ..............  caractère dans X
CC61  JSR $F4Ø9    ..............  et appel de la routine moniteur
CC64  PLA          ..............
CC65  TAX          ..............  on reprend X
CC66  PLA          ..............
CC67  TAY          ..............  et Y
CC68  LDA 27       ..............  et A
CC6A  AND #FF      ..............  et on positionne N et Z
CC6C  RTS          ..............
CC6D ....  BYT #ØD,#ØA
CC6F ....  BYT '?PRINTER ERROR'
CC7D ....  BYT #ØD,#ØA,#ØØ
```

GESTION D'UNE HYPOTHETIQUE VIDEO INVERSE

Bogue: tout avait été prévu pour gérer la vidéo inverse, les mots clé (INVERSE et NORMAL) devaient surement sauter ici. C'était sans compter le fait que #F4Ø9 élimine b7. En fait, le plus décourageant a sans doute été la gestion du clignotement du curseur, qui devient (relativement) plus compliquée. Du reste, il est amusant de poker n'importe quoi en #2F7.

Le vice a été poussé jusqu'à initialiser #2F7 lors d'un RESET !

'INVERSE' (COMMANDE AVORTEE)

```
CC8Ø  LDA #8Ø      ..............  prendre code pour video inverse (b7=1)
CC82  BYT #2C      ..............  et sauter l'instruction suivante
```

'NORMAL' (COMMANDE AVORTEE)

```
CC83  LDA #00     ..............  prendre code pour vidéo normale (b7=0)
CC85  STA 02F7    ..............  et sauver dans drapeau vidéo
CC88  RTS         ..............

..............  CCEE  BIT 02F1   sortie sur imprimante ?
..............  CCF1  BPL CCFB   non, sauter
..............  CCF3  PHA       sauver le code
..............  CCF4  JSR $023E  et l'imprimer via ce vecteur (#F5CB)
..............  CCF7  PLA       récupérer le code
..............  CCF8  AND #FF   et positionner N et Z
..............  CCFA  RTS

..............  CCFB  STX 27     sauver X
..............  CCFD  TAX       et mettre le caractère dans X
..............  CCFE  JSR $F77C  et appeler impréssion moniteur
..............  CD01  CMP #20   est-ce un caractère de controle ?
..............  CD03  BCC CD09  oui, placer tabulation
..............  CD05  CMP #7F   est-ce DEL ?
..............  CD07  BNE CD0E  non, finir
..............  CD09  LDX 0269  prendre colonne curseur
..............  CD0C  STX 30    et ajuster colonne tabulation
..............  CD0E  LDX 27    reprendre X
..............  CD10  AND #FF   et positionner N et Z
..............  CD12  RTS
```

'!' (COMMANDE)

```
CC89  JMP (02F5)  CD13  JMP (02F5)  continuer à l'adresse utilisateur...
```

'TRON' (COMMANDE)

Irritant: sur la V1.0,le drapeau est remis à 0 à chaque Ready

Programmation: SEC:BYT #24:CLC:ROR 02F4 eut été plus judicieux.

```
CC8C  LDA #80     CD16  LDA #80     mettre drapeau TRON
CC8E  BYT #2C     CD18  BYT #2C     et sauter instruction suivante
```

'TROFF' (COMMANDE)

```
CC8F  LDA #00     CD19  LDA #00     mettre drapeau TROFF
CC91  STA 02F4    CD1B  STA 02F4    et le sauver
CC94  RTS         CD1E  RTS
```

Traiter les erreurs de READ/GET/INPUT

Le READ arrive ici lorsqu'après une variable, il ne trouve ni une ',' (autre donnée à lire) ni la fin de l'instruction.

```
CC95  LDA 2C      CD1F  LDA 2C      prendre indicateur
CC97  BEQ CCAA    CD21  BEQ CD36    sauter si INPUT
CC99  BMI CC9F    CD23  BMI CD29    sauter si READ
CC9B  LDY #FF     CD25  LDY #FF     GET: simuler mode direct
CC9D  BNE CCA3    CD27  BNE CD2D    inconditionnel: erreur
CC9F  LDA AE      CD29  LDA AE      READ
CCA1  LDY AF      CD2B  LDY AF      récupérer la ligne du READ
CCA3  STA A8      CD2D  STA A8
CCA5  STY A9      CD2F  STY A9      qui est le vrai numéro courant
.............     CD31  LDX #A8     'TYPE MISMATCH ERROR' pour V1.1
CCA7  JMP $CFE4   CD33  JMP $C47E   ET 'SYNTAX ERROR ' pour V1.0
CCAA  LDA #F9     CD36  LDA #85     INPUT
CCAC  LDY #CD     CD38  LDY #CE     AY pointe sur '?REDO FROM START'
CCAE  JSP $CBED   CD3A  JSR $CCB0   et afficher le message
CCB1  LDA AC      CD3D  LDA AC
CCB3  LDY AD      CD3F  LDY AD      récupérer TXTPTR
CCB5  STA E9      CD41  STA E9
CCB7  STY EA      CD43  STY EA      et recommencer le INPUT
CCB9  RTS         CD45  RTS
```

'GET' (COMMANDE)

Principe: le GET est traité comme le READ. Le pointeur de DATA est placé en #0036, où on place un 0. Lors du READ, il va être détecté une donnée vide, qui sera traitée différement pour le GET: on ira à ce moment là chercher un caractère au clavier pour le mettre en #0035 et y lire la donnée. Il faut ajuster 'manuellement TXTPTR à #0035 car si on frappe la touche espace, elle sera ignorée par la routine #00E8.

Comme d'habitude,le fait de passer par READ permet d'uniformiser le traitement de plusieurs instructions et donc de raccourcir leur traitement.

Bogue: sur la V1.Ø, l'apostrophe n'étant pas gérée correctement, elle déclenchera une erreur de syntaxe, toujours à cause du JSR #ØØE8.

```
CCBA  JSR $D419    CD46  JSR $D4D2    vérifier mode programme
CCBD  LDX #36      CD49  LDX #36      indiquer début de la donnée
CCBF  LDY #ØØ      CD4B  LDY #ØØ      et marquer la fin de l'entrée
CCC1  STY 36       CD4D  STY 36       par un ØØ
CCC3  LDA #4Ø      CD4F  LDA #4Ø      code pour GET (non nul et b7=Ø, b6=1)
CCC5  JSR $CDØ3    CD51  JSR $CD8F    faire READ
CCC8  RTS          CD54  RTS          JMP aurait bien fait l'affaire
```

'INPUT' (COMMANDE)

Principe: INPUT est ramené à un READ dont la donnée à lire serait dans le tampon clavier. La seule particularité est donc le traitement des erreurs (EXTRA IGNORED etc...)

Astuce: au sortir de la mise en place de l'ordre, Y vaut #ØØ. Ce qui correspond au code du INPUT. On saute alors sur le deuxième octet du LDA #98 qui est en fait un TYA. Bien joué !

```
CCC9  LSR 2E       CD55  LSR 2E       inhiber le Ctrl O
CCCB  CMP #'"'     CD57  CMP #'"'     INPUT suivi d'un message ?
CCCD  BNE CCDA     CD59  BNE CD66     non, on saute
CCCF  JSR $CF99    CD5B  JSR $DØ25    oui, on évalue la chaine
CCD2  LDA #';'     CD5E  LDA #';'
CCD4  JSR $CFDB    CD6Ø  JSR $DØ67    demander un ';'
CCD7  JSR $CBFØ    CD63  JSR $CCB3    afficher la chaine
CCDA  JSR $D419    CD66  JSR $D4D2    vérifier mode programme
CCDD  LDA #','     CD69  LDA #','     prendre virgule
CCDF  STA 34       CD6B  STA 34       pour simuler un séparateur de DATA
CCE1  LDA #ØØ      CD6D  LDA #ØØ
CCE3  STA 17       CD6F  STA 17       initialiser mode Ctrl C
CCE5  JSR $CCF4    CD71  JSR $CD8Ø    remplir le tampon (YX vaut #ØØ34)
CCE8  LDA 35       CD74  LDA 35       une donnée a été entrée ?
CCEA  BNE CDØ2     CD76  BNE CD8E     oui, OK
CCEC  LDA 17       CD78  LDA 17       non, était-ce à cause de Ctrl C ?
CCEE  BEQ CCE1     CD7A  BEQ CD6D     non, on recommence l'entrée
CCFØ  CLC          CD7C  CLC          oui, on fait BREAK
CCF1  JMP $C94E    CD7D  JMP $C98Ø
CCF4  JSR $CC1Ø    CD8Ø  JSR $CCD7    afficher '?'
CCF7  JSR $CCØD    CD83  JSR $CCD4    afficher ' '
CCFA  JMP $C5A2    CD86  JMP $C592    et prendre un ordre dans le tampon
```

Principe: la commande READ est en fait appelée aussi par le GET et INPUT. L'octet #2C permet de savoir quelle est la commande en cours: il vaut ØØ (nul, b7=Ø et b6=Ø) pour INPUT ,#4Ø pour GET (non nul, b7=Ø et b6=1) et #98 pour READ (b7=1 et non nul).

On utilise un pseudo pointeur de DATA (#B2-3) qui est initialisé sur le vrai pointeur pour READ, sur #ØØ34 pour INPUT et sur #ØØ36 pour GET. A la fin, s'il s'agit d'un READ, le vrai pointeur (#BØ-#B1) sera replacé à sa valeur courante.

Il faut aussi sauver (en #BA-#BB) la valeur de TXTPTR car il va servir pour évaluer les données. Il sera dans tous les cas récupéré à la fin

Les valeurs numériques et alphanumériques sont traitées à part. Une complication pour les valeurs alphanumériques: si elles sont entourées de guillemets (seule utilité: permettre des espaces au début de la donnée qui seraient sinon ignorés, ou pouvoir entrer des virgules sans qu'elles soient considérées comme des séparateurs), il va falloir les sauter, et il s'en suit aussi quelques astuces concernant les séparateurs à appliquer.

```
CCFD  LDX BØ      CD89  LDX BØ      prendre adresse du pointeur
CCFF  LDY B1      CD8B  LDY B1      de DATA
CDØ1  LDA #98     CD8D  LDA #98     b7=1: indiquer READ et sauter le TYA
CDØ2  TYA         CD8E  TYA         #98 est le code de TYA !
CDØ3  STA 2C      CD8F  STA 2C      placer l'indicateur de commande
CDØ5  STX B2      CD91  STX B2      sauver le pointeur de DATA
CDØ7  STY B3      CD93  STY B3
CDØ9  JSR $DØFC   CD95  JSR $D188   prendre la première variable
CDØC  STA B8      CD98  STA B8
CDØE  STY B9      CD9A  STY B9      et sauver son adresse
CD1Ø  LDA E9      CD9C  LDA E9
CD12  LDY EA      CD9E  LDY EA
CD14  STA BA      CDAØ  STA BA
CD16  STY BB      CDA2  STY BB      sauver aussi TXTPTR
CD18  LDX B2      CDA4  LDX B2
CD1A  LDY B3      CDA6  LDY B3      récupérer le pointeur de DATA
CD1C  STX E9      CDA8  STX E9
CD1E  STY EA      CDAA  STY EA      dans TXTPTR
CD2Ø  JSR $ØØE8   CDAC  JSR $ØØE8   prendre caractère de la donnée
CD23  BNE CD42    CDAF  BNE CDCE    si pas vide,sauter

Traiter une donnée vide

CD25  BIT 2C      CDB1  BIT 2C      au fait,c'était quelle instruction ?
CD27  BVC CD36    CDB3  BVC CDC2    sauter si pas GET
```

```
CD29  JSR $E9Ø5   CDB5  JSR $EB78
CD2C  BPL CD29    CDB8  BPL CDB5    attendre un caractère
CD2E  STA 35      CDBA  STA 35      et le sauver (il y a un Ø en #36)
CD3Ø  LDX #34     CDBC  LDX #34
CD32  LDY #ØØ     CDBE  LDY #ØØ     prendre adresse de la donnée
CD34  BEQ CD3E    CDCØ  BEQ CDCA    inconditionnel: lire la donnée
CD36  BMI CDA9    CDC2  BMI CE35    sauter si READ pour chercher DATA suivante
CD38  JSR $CC1Ø   CDC4  JSR $CCD7   c'est donc INPUT: afficher ?
CD3B  JSR $CCF4   CDC7  JSR $CD8Ø   reprendre les données au clavier
CD3E  STX E9      CDCA  STX E9
CD4Ø  STY EA      CDCC  STY EA      et ajuster TXTPTR en #ØØ34

CD42  JSR $ØØE2   CDCE  JSR $ØØE2   ajuster TXTPTR après la virgule
CD45  BIT 28      CDD1  BIT 28      tester type demandé
CD47  BPL CD7A    CDD3  BPL CEØ6    si numérique, on saute

Traiter alphanumérique demandé

CD49  BIT 2C      CDD5  BIT 2C      type alpha:
CD4B  BVC CD56    CDD7  BVC CDE2    si pas GET, on saute
CD4D  INX         CDD9  INX         X=#35
CD4E  STX E9      CDDA  STX E9      ajuster TXTPTR (on a pu rentrer un ' '
CD5Ø  LDA #ØØ     CDDC  LDA #ØØ     qui est ignoré par #ØØE2)
CD52  STA 24      CDDE  STA 24      indiquer terminateur=ØØ
CD54  BEQ CD62    CDEØ  BEQ CDEE    inconditionnel
CD56  STA 24      CDE2  STA 24      mettre premier caractère comme terminateur
CD58  CMP #'"'    CDE4  CMP #'"'    au fait, est-ce un '"' ?
CD5A  BEQ CD63    CDE6  BEQ CDEF    oui,c'était un bon terminateur,on saute,C=1
CD5C  LDA #':'    CDE8  LDA #':'    non, le séparateur d'instruction
CD5E  STA 24      CDEA  STA 24      est un meilleur séparateur
CD6Ø  LDA #','    CDEC  LDA #','    mais la virgule est prioritaire
CD62  CLC         CDEE  CLC         indiquer pas de caractère à sauter
CD63  STA 25      CDEF  STA 25      sauver deuxième terminateur
CD65  LDA E9      CDF1  LDA E9
CD67  LDY EA      CDF3  LDY EA      prendre TXTPTR
CD69  ADC #ØØ     CDF5  ADC #ØØ     pour lui ajouter 1 si l'entrée
CD6B  BCC CD6E    CDF7  BCC CDFA    commance par un guillemet
CD6D  INY         CDF9  INY         sans oublier le poids fort
CD6E  JSR $D5ØØ   CDFA  JSR $D5BB   évaluer la chaine pointée par AY
CD71  JSR $D852   CDFD  JSR $D9ØD   ajuster TXTPTR sur la chaine
CD74  JSR $CBØ7   CEØØ  JSR $CB51   faire LET
CD77  JMP $CD82   CEØ3  JMP $CEØE   passer à la suite
```

```
Traiter numérique demandé

CD7A  JSR $DFCF   CE06  JSR $DFE7   évaluer une valeur numérique
CD7D  LDA 29      CE09  LDA 29      prendre drapeau entier/réel
CD7F  JSR $CAEF   CE0B  JSR $CB39   et faire LET

Finir un READ

CD82  JSR $00E8   CE0E  JSR $00E8   prendre caractère courant (dans les DATA)
CD85  BEQ CD8E    CE11  BEQ CE1A    et sauter si fin d'instruction
CD87  CMP #','    CE13  CMP #','    est-ce un séparateur de DATA ?
CD89  BEQ CD8E    CE15  BEQ CE1A    oui, c'est bon
CD8B  JMP $CC95   CE17  JMP $CD1F   non, il s'agit donc d'une erreur
CD8E  LDA E9      CE1A  LDA E9
CD90  LDY EA      CE1C  LDY EA      sauver le pointeur de DATA
CD92  STA B2      CE1E  STA B2      qui pointe sur la ',' ou le #00
CD94  STY B3      CE20  STY B3
CD96  LDA BA      CE22  LDA BA      récupérer TXTPTR
CD98  LDY BB      CE24  LDY BB
CD9A  STA E9      CE26  STA E9
CD9C  STY EA      CE28  STY EA      pour continuer le READ
CD9E  JSR $00E8   CE2A  JSR $00E8   plus de variable à charger ?
CDA1  BEQ CDCF    CE2D  BEQ CE5B    non, on sort
CDA3  JSR $CFD9   CE2F  JSR $D065   oui on veut une ','
CDA6  JMP $CD09   CE32  JMP $CD95   et on recommence

Chercher DATA suivante

CDA9  JSR $CA1C   CE35  JSR $CA4E   trouver la fin de l'instruction
CDAC  INY         CE38  INY         et sauter
CDAD  TAX         CE39  TAX         le séparateur était-il un ':'
CDAE  BNE CDC2    CE3A  BNE CE4E    oui, on saute
CDB0  LDX #2A     CE3C  LDX #2A     préparer X pour 'OUT OF DATA'
CDB2  INY         CE3E  INY         indexer le poids fort du lien
CDB3  LDA (E9),Y  CE3F  LDA (E9),Y  et prendre le lien
CDB5  BEQ CE20    CE41  BEQ CEAC    si fin du programme, 'OUT OF DATA'
CDB7  INY         CE43  INY         sinon indexer le numéro de ligne
CDB8  LDA (E9),Y  CE44  LDA (E9),Y  prendre le poids faible
CDBA  STA AE      CE46  STA AE      et le sauver
CDBC  INY         CE48  INY
CDBD  LDA (E9),Y  CE49  LDA (E9),Y  prendre le poids fort
CDBF  INY         CE4B  INY
CDC0  STA AF      CE4C  STA AF      et le sauver
CDC2  LDA (E9),Y  CE4E  LDA (E9),Y  prendre code premier mot-clé de la ligne
```

```
CDC4  TAX          CE5Ø  TAX          et le sauver dans X
CDC5  JSR $CAØD    CE51  JSR $CA3F    ajuster TXTPTR au début de la ligne
CDC8  CPX #&DATA   CE54  CPX #&DATA   la ligne commence-t-elle par DATA ?
CDCA  BNE CDA9     CE56  BNE CE35     non, continuer la recherche
CDCC  JMP $CD42    CE58  JMP $CDCE    oui, aller affecter la donnée

Sortir d'un READ/GET/INPUT

CDCF  LDA B2       CE5B  LDA B2       prendre pointeur de DATA courant
CDD1  LDY B3       CE5D  LDY B3
CDD3  LDX 2C       CE5F  LDX 2C       au fait, quelle instruction ?
CDD5  BPL CDDA     CE61  BPL CE66
CDD7  JMP $C929    CE63  JMP $C95C    si c'est READ, replacer #BØ-#B1 et sortir
CDDA  LDY #ØØ      CE66  LDY #ØØ      GET ou INPUT
CDDC  LDA (B2),Y   CE68  LDA (B2),Y   prendre caractère de la donnée
CDDE  BEQ CDE7     CE6A  BEQ CE73     et sauter si c'est la fin (tjs pour GET)
CDEØ  LDA #E8      CE6C  LDA #74
CDE2  LDY #CD      CE6E  LDY #CE      AY pointe sur '?EXTRA IGNORED'
CDE4  JMP $CBED    CE7Ø  JMP $CCBØ    afficher le message et finir
CDE7  RTS          CE73  RTS

CDE8 CE74  BYT '?EXTRA IGNORED'
CDF6 CE82  BYT #ØD,#ØA,#ØØ
CDF9 CE85  BYT '?REDO FROM START'
CE09 CE95  BYT #ØD,#ØA,#ØØ
```

'NEXT' (COMMANDE)

Principe: après avoir trouvé le bloc correspondant à l'indice, on ajoute le pas à la valeur courante de l'indice. Si on dépasse la valeur limite entrée lors du FOR, on sort et on continue s'il y d'autres indices.

La sortie se fait par un saut à l'interpréteur car l'adresse de retour n'est pas empilée (elle est éliminée par le FOR). De plus, lorsqu'on arrive au NEXT, cette adresse de retour est toujours sur la pile. Si on doit éliminer un bloc, puis repartir au début, il n'y aura pas d'adresse sur la pile, et on risque de se décaler de 2 octets. C'est pourquoi on recommence par un JSR et non un JMP pour placer 2 octets au sommet de la pile.

```
CE0C  BNE CE12     CE98  BNE CE9E     sauter si un indice est précisé
CE0E  LDY #ØØ      CE9A  LDY #ØØ      sinon poids fort de l'adresse de l'indice
CE10  BEQ CE15     CE9C  BEQ CEA1     inconditionnel: est mis à Ø
CE12  JSR $DØFC    CE9E  JSR $D188    prendre adresse de l'indice
CE15  STA B8       CEA1  STA B8
```

```
CE17  STY B9      CEA3  STY B9      et la stocker
CE19  JSR $C3C4   CEA5  JSR $C3C6  rechecher un bloc 'FOR' sur la pile
CE1C  BEQ CE22    CEA8  BEQ CEAE   sauter si trouvé
CE1E  LDX #00     CEAA  LDX #00     sinon,X indexe 'NEXT WITHOUT FOR'
CE20  BEQ CE88    CEAC  BEQ CF14    et imprimer l'erreur
CE22  TXS         CEAE  TXS         placer S sur le bon bloc
CE23  TXA         CEAF  TXA         puis calculer le poids faible
CE24  CLC         CEB0  CLC         de l'adresse du pas d'incrémentation
CE25  ADC #04     CEB1  ADC #04     (qui est dans la pile
CE27  PHA         CEB3  PHA         et le sauver
CE28  ADC #06     CEB4  ADC #06     calculer aussi l'index
CE2A  STA 93      CEB6  STA 93      pour la valeur limite
CE2C  PLA         CEB8  PLA         AY=#01XX
CE2D  LDY #01     CEB9  LDY #01
CE2F  JSR $DE73   CEBB  JSR $DE7B   (AY) --> ACC1
CE32  TSX         CEBE  TSX
CE33  LDA 0109,X  CEBF  LDA 0109,X  récupérer signe du pas
CE36  STA D5      CEC2  STA D5      et le sauver
CE38  LDA B8      CEC4  LDA B8
CE3A  LDY B9      CEC6  LDY B9      prendre l'adresse de l'indice
CE3C  JSR $DA97   CEC8  JSR $DB22   (AY)+ACC1 --> ACC1:Calculer nouvelle valeur
CE3F  JSR $DEA1   CECB  JSR $DEA9   ACC1 --> (#B8-#B9) et placer nouvel indice
CE42  LDY #01     CECE  LDY #01     #93-Y=#01xx adresse de la valeur limite
CE44  JSR $DF36   CED0  JSR $DF4E   comparer ACC1 et (AY)
CE47  TSX         CED3  TSX
CE48  SEC         CED4  SEC         le signe de la comparaison
CE49  SBC 0109,X  CED5  SBC 0109,X  doit être le même que celui du pas (STEP)
CE4C  BEQ CE65    CED8  BEQ CEF1    oui, la boucle est finie

Recommencer la boucle

CE4E  LDA 010F,X  CEDA  LDA 010F,X  non, on repart:
CE51  STA A8      CEDD  STA A8      on récupère le numéro de ligne poids faible
CE53  LDA 0110,X  CEDF  LDA 0110,X
CE56  STA A9      CEE2  STA A9      et aussi poids fort
CE58  LDA 0112,X  CEE4  LDA 0112,X
CE5B  STA E9      CEE7  STA E9      puis TXTPTR poids faible
CE5D  LDA 0111,X  CEE9  LDA 0111,X
CE60  STA EA      CEEC  STA EA      et aussi poids fort
CE62  JMP $C8AD   CEEE  JMP $C8C1   et on saute à l'interpréteur

Sortie de boucle

CE65  TXA         CEF1  TXA         ajuster la pile au dessus
```

```
CE66  ADC #11     CEF2  ADC #11     du bloc FOR à suprimer (C=1, ADC #12 donc)
CE68  TAX         CEF4  TAX
CE69  TXS         CEF5  TXS
CE6A  JSR $00E8   CEF6  JSR $00E8   prendre caractère courant
CE6D  CMP #','    CEF9  CMP #','    y-a-t-il plusieurs indices ?
CE6F  BNE CE62    CEFB  BNE CEEE    non, on sort
CE71  JSR $00E2   CEFD  JSR $00E2   oui,on saute la virgule
CE74  JSR $CE12   CF00  JSR $CE9E   et on repart,en plaçant une adresse bidon
```

EVALUER UNE EXPRESSION NUMERIQUE

Entrée: TXTPTR.

Sortie: le nombre est dans ACC1.
A, N et Z positionnés selon son exposant.

```
CE77  JSR $CE8B   CF03  JSR $CF17   Evaluer l'expression
```

VERIFIER VARIABLE NUMERIQUE

```
CE7A  CLC         CF06  CLC         Indiquer numérique demandé
CE7B  BYT #24     CF07  BYT #24     sauter instruction suivante
```

VERIFIER VARIABLE ALPHANUMERIQUE

```
CE7C  SEC         SEC               indiquer chaine demandée
CE7D  BIT 28      CF09  BIT 28      tester drapeau numérique/chaine
CE7F  BMI CE84    CF0B  BMI CF10    si chaine ,sauter
CE81  BCS CE86    CF0D  BCS CF12    si numérique et chaine demandée, erreur
CE83  RTS         CF0F  RTS
CE84  BCS CE83    CF10  BCS CF0F    si chaine et chaine demandée, OK
CE86  LDX #A8     CF12  LDX #A8     indexer le message 'TYPE MISMATCH'
CE88  JMP $C485   CF14  JMP $C47E   et déclencher l'erreur
```

EVALUER UNE EXPRESSION

Entrée: TXTPTR bien placé.

Sortie: la valeur est dans ACC1 (numérique). Pour une chaine aussi (adresse du descripteur dans #D3-#D4).

#28=#ØØ (numérique) ou #FF (chaine).

A,N et Z sont placés selon l'exposant (résultat numérique) ou la longueur de la chaine.

Principe: on évalue tour à tour tous les opérandes, suivi des opérateurs. Selon la priorité, ceux-ci sont exécutés, ou l'opérande est empilé,ainsi que sa priorité et son adresse (-1).

L'exécution d'un opérande reviens à dépiler une valeur et la mettre dans ACC2 (opérande de gauche), à placer correctement A et Z ,et à sortir par RTS, puisque l'adresse a été empilée.

La valeur de priorité empilée au début est #ØØ, qui sert de butée à la pile des opérateurs. Il faut une autre butée pour connaitre la fin de l'expréssion (plus d'opérateur), qui ne correspond pas nécéssairement à la fin de l'évaluation, car il peu rester des opérateurs à exécuter sur la pile. La variable #BA, qui contient #FF (fin) ou l'index de l'opérateur, qui est toujours positif, tient ce role de 'fin physique'.

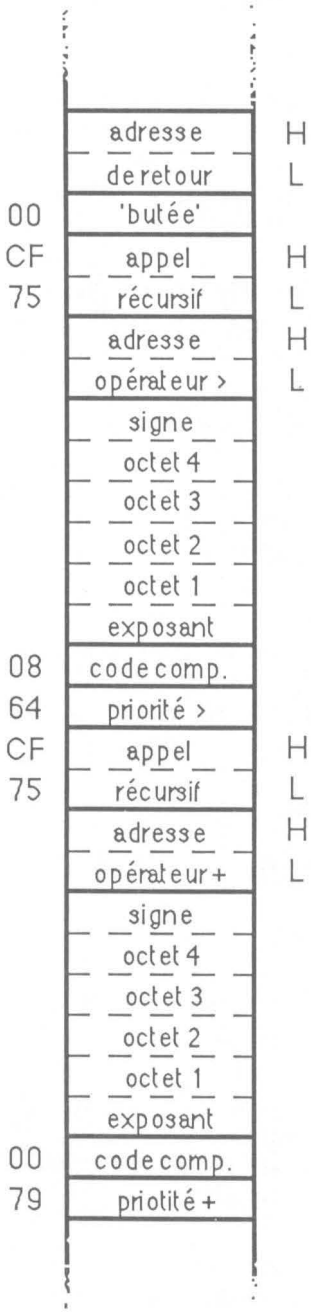
Lorsqu'on appelle la routine qui ramasse un nombre ou une chaine (#CF74/#DØØØ), celle-ci commence par sauter un caractère, qui est normalement l'opérateur. C'est pourquoi il faut ajuster TXTPTR au début et lors de l'évaluation d'une condition.

Pour bien comprendre cette routine, qui est de loin la plus compliquée, de la ROM, on peut se reporter à son organigramme

Sa difficulté vient de son aspect récursif, où la pile joue un role déterminant. Pour bien comprendre ce problème, se reporter à l'exemple donné.

FIGURE CE-B et CE-A

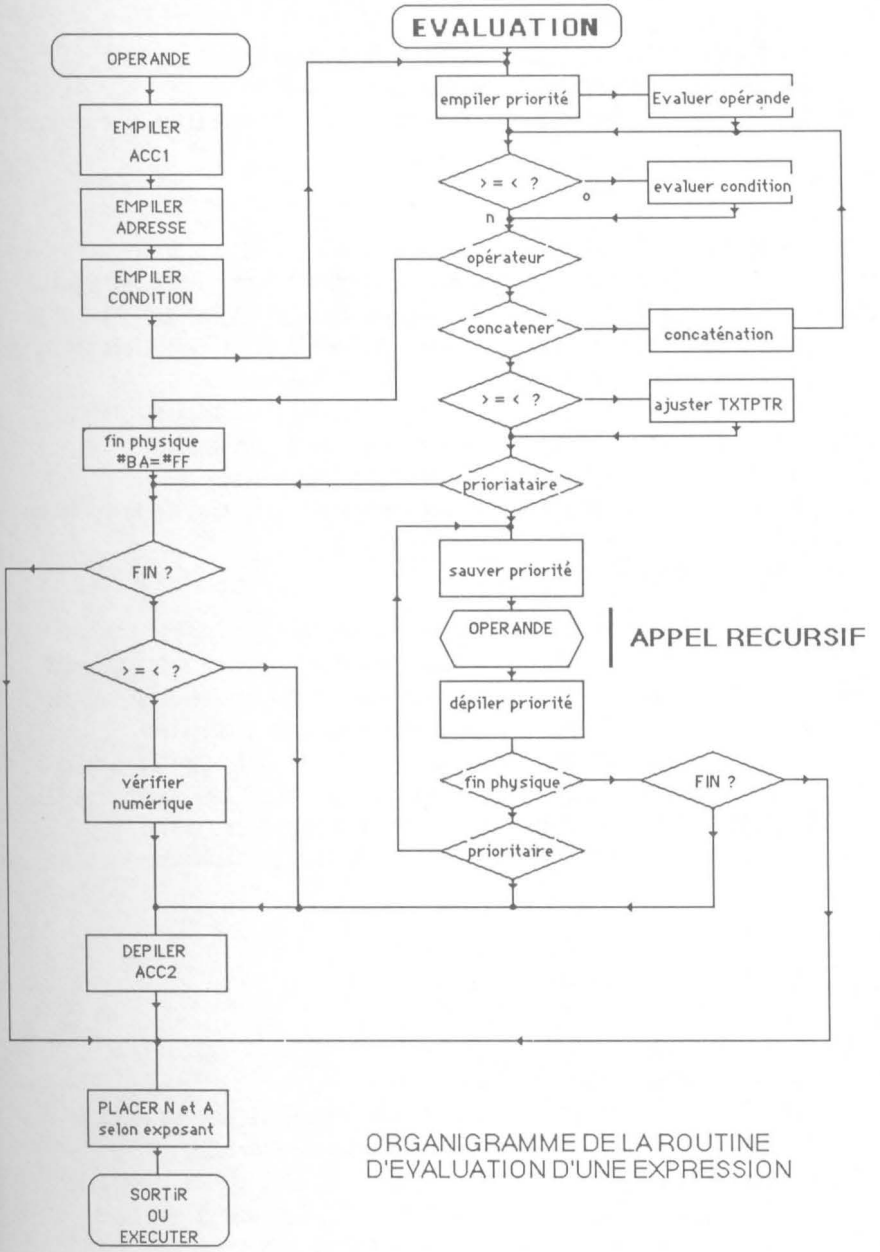
```
CE8B  LDX E9      CF17  LDX E9      Décrémenter TXTPTR
CE8D  BNE CE91    CF19  BNE CF1D    pour ajuster
CE8F  DEC EA      CF1B  DEC EA      à cause du JSR #CF74/#DØØØ
CE91  DEC E9      CF1D  DEC E9
CE93  LDX #ØØ     CF1F  LDX #ØØ     priorité initiale=Ø (plus d'opérateur)
CE95  BYT #24     CF21  BYT #24     sauter instruction suivante

CE96  PHA         CF22  PHA         empiler code de comparaison (ou Ø)
CE97  TXA         CF23  TXA         sauver la priorité courante
CE98  PHA         CF24  PHA         toujours au sommet de la pile
CE99  LDA #Ø1     CF25  LDA #Ø1     demander 2 octets sur la pile
CE9B  JSR $C43B   CF27  JSR $C437
CE9E  JSR $CF74   CF2A  JSR $DØØØ   Evaluer un opérande dans ACC1
CEA1  LDA #ØØ     CF2D  LDA #ØØ     indiquer pas d'opérateur de condition
CEA3  STA BC      CF2F  STA BC      pour l'instant...
CEA5  JSR $ØØE8   CF31  JSR $ØØE8   prendre caractère courant (opérateur)
```



Etat de la pile lors de l'évaluation de l'expression A > B + C

Les adresses sont données pour la V1.1

Figure CE-B



ORGANIGRAMME DE LA ROUTINE D'EVALUATION D'UNE EXPRESSION

Figure CE-A

```
CEA8  SEC          CF34  SEC          inutile
CEA9  SBC #&)      CF35  SBC #&)
CEAB  BCC CEC4     CF37  BCC CF5Ø     sauter si code supérieur à &)
CEAD  CMP #Ø3      CF39  CMP #Ø3      sinon comparer à &( (#D3+3)
CEAF  BCS CEC4     CF3B  BCS CF5Ø     et sauter,C=1,si pas opérateur de condition

Evaluer code de comparaison

CEB1  CMP #Ø1      CF3D  CMP #Ø1      est-ce &= ? A=Ø ()) ,1 (=) ,2 (()
CEB3  ROL A        CF3F  ROL A        C=Ø si &) :A=Ø ()),2 (=),4 (() :Bits Ø,1,2
CEB4  EOR #Ø1      CF4Ø  EOR #Ø1      et inverser la relation car Bit actif à 1
CEB6  EOR BC       CF42  EOR BC       inverser si un bit etait dejà placé
CEB8  CMP BC       CF44  CMP BC       comparer à l'ancien masque:
CEBA  BCC CF1D     CF46  BCC CFA9     si c'est inférieur, redondance:'SYNTAX'
CEBC  STA BC       CF48  STA BC       sauver le masque de comparaison
CEBE  JSR $ØØE2    CF4A  JSR $ØØE2    prendre caractère suivant
CEC1  JMP $CEA8    CF4D  JMP $CF34    et recommencer l'évaluation de la condition

Traiter code opérateur ou autre...

CEC4  LDX BC       CF5Ø  LDX BC       prendre condition
CEC6  BNE CEF4     CF52  BNE CF8Ø     si une condition, exécuter selon priorité
CEC8  BCS CF49     CF54  BCS CFD5     pas opérateur: exécuter opérateur sur pile
CECA  ADC #Ø7      CF56  ADC #Ø7      retrouver le code de l'opérateur
CECC  BCC CF49     CF58  BCC CFD5     pas opérateur: exécuter opérateur sur pile
CECE  ADC 28       CF5A  ADC 28       ajouter drapeau chaine (#FF ou Ø) et 1 (C)
CEDØ  BNE CED5     CF5C  BNE CF61     ce qui donne Ø pour &+ et chaine
CED2  JMP $D6AC    CF5E  JMP $D767    si tel est le cas, concaténation

Traiter un opérateur

CED5  ADC #FF      CF61  ADC #FF      code-1 (Ø à 6)
CED7  STA 91       CF63  STA 91       et le sauver
CED9  ASL A        CF65  ASL A        *2
CEDA  ADC 91       CF66  ADC 91       +ancien=*3
CEDC  TAY          CF68  TAY          dans Y pour indéxer adresse et priorité
CEDD  PLA          CF69  PLA          récupérer priorité courante
CEDE  CMP CØCC,Y   CF6A  CMP CØCC,Y   et comparer à la priorité de l'opérateur
CEE1  BCS CF4E     CF6D  BCS CFDA     si celle-ci >=, appliquer l'opérateur
CEE3  JSR $CE7A    CF6F  JSR $CFØ6    interdire les chaines
CEE6  PHA          CF72  PHA          sauver la priorité
CEE7  JSR $CFØD    CF73  JSR $CF99    et évaluer un nouvel opérande (récursif)
CEEA  PLA          CF76  PLA          récupérer la priorité
CEEB  LDY BA       CF77  LDY BA       index de l'opérateur (ou #FF pas de reste)
```

```
CEED  BPL CF06    CF79  BPL CF92    si encore opérateur sur pile, exécuter
CEEF  TAX         CF7B  TAX         placer Z selon priorité
CEF0  BEQ CF4C    CF7C  BEQ CFD8    fin d'évaluation: sortir
CEF2  BNE CF57    CF7E  BNE CFE3    sinon, appliquer opérateur restant

Traiter )=(

CEF4  LSR 28      CF80  LSR 28      prendre dans C drapeau  chaine/numérique
CEF6  TXA         CF82  TXA         prendre code de la condition
CEF7  ROL A       CF83  ROL A       et y incorporer le drapeau chaine
CEF8  LDX E9      CF84  LDX E9
CEFA  BNE CEFE    CF86  BNE CF8A    décrémenter TXTPTR
CEFC  DEC EA      CF88  DEC EA
CEFE  DEC E9      CF8A  DEC E9
CF00  LDY #1B     CF8C  LDY #1B     Y=index pour (=), opérateur No 9
CF02  STA BC      CF8E  STA BC      sauver condition+drapeau chaine
CF04  BNE CEDD    CF90  BNE CF69    inconditionnel: exécuter ou non opérateur

Exécuter ou non opérateur sur la pile

CF06  CMP C0CC,Y  CF92  CMP C0CC,Y  comparer à priorité de l'opérateur
CF09  BCS CF57    CF95  BCS CFE3    si supérieur, exécuter l'opérateur
CF0B  BCC CEE6    CF97  BCC CF72    sinon, on recommence

Placer sur la pile adresse de l'opérande et évaluer autre opérande

CF0D  LDA C0CE,Y  CF99  LDA C0CE,Y  prendre adresse d'éxécution poids fort
CF10  PHA         CF9C  PHA         et la sauver comme adresse de retour
CF11  LDA C0CD,Y  CF9D  LDA C0CD,Y
CF14  PHA         CFA0  PHA         idem poids faible
CF15  JSR $CF20   CFA1  JSR $CFAC   empiler l'opérande de gauche
CF18  LDA BC      CFA4  LDA BC      prendre code de condition (ou 0)
CF1A  JMP $CE96   CFA6  JMP $CF22   et recommencer l'évaluation

CF1D  JMP $CFE4   CFA9  JMP $D07B   Exécuter un 'SYNTAX ERROR'
```

EMPILER UN OPERANDE

```
Entrée: Y=index de l'opérateur (3#0,...,6)
Sortie: X=priorité de l'opérateur,AACC1 sur la pile (signe puis 5 octets)

CF20  LDA D5      CFAC  LDA D5      prendre dans A le signe de ACC1
CF22  LDX C0CC,Y  CFAE  LDX C0CC,Y  et dans X la priorité
```

```
CF25  TAY          CFB1  TAY          et sauver dans Y le signe
CF26  PLA          CFB2  PLA          récupérer adresse de retour poids faible
CF27  STA 91       CFB3  STA 91       et la sauver
CF29  PLA          CFB5  PLA          idem poids fort
CF2A  STA 92       CFB6  STA 92       la sauver aussi
CF2C  INC 91       CFB8  INC 91       et l'incrémenter
CF2E  BNE CF32     CFBA  BNE CFBE     pour l'ajuster
CF30  INC 92       CFBC  INC 92       (un JSR empile l'adresse de retour -1)
CF32  TYA          CFBE  TYA          récupérer signe
CF33  PHA          CFBF  PHA          sauver le signe
```

EMPILER AACC1

Entrée: $91-2 doit contenir l'adresse de retour

Sortie: AACC1 empilé

```
CF34  JSR $DEEC    CFC0  JSR $DEF4   Arrondir ACC1 fonction du Bit d'extension
CF37  LDA D4       CFC3  LDA D4       et empiler ACC1
CF39  PHA          CFC5  PHA          Octet 4
CF3A  LDA D3       CFC6  LDA D3
CF3C  PHA          CFC8  PHA          Octet 3
CF3D  LDA D2       CFC9  LDA D2
CF3F  PHA          CFCB  PHA          Octet 2
CF40  LDA D1       CFCC  LDA D1
CF42  PHA          CFCE  PHA          Octet 1
CF43  LDA D0       CFCF  LDA D0
CF45  PHA          CFD1  PHA          Et exposant
CF46  JMP (0091)   CFD2  JMP (0091)   et retourner d'où on vient...
```

EXECUTER UN OPERATEUR

Remarque: cette routine finissant par un LDA D0, A et Z sont positionnés selon l'exposant lors de l'exécution d'un opérateur. Il est de même nécessaire au fonctionnement correct des opérateurs que le produit des signes soit correctement placé. Tout ceci est inutile pour les fonctions.

```
CF49  LDY #FF      CFD5  LDY #FF      indiquer dernier opérateur
CF4B  PLA          CFD7  PLA          récupère priorité
CF4C  BEQ CF71     CFD8  BEQ CFFD     si fin de l'expression, sortir

CF4E  CMP #64      CFDA  CMP #64      Est-ce un opérateur de condition?
```

```
CF50  BEQ CF55    CFDC  BEQ CFE1    oui, OK
CF52  JSR $CE7A   CFDE  JSR $CF06   non, vérifier numérique
CF55  STY BA      CFE1  STY BA      et sauver l'index de l'opérateur (ou #FF)

CF57  PLA         CFE3  PLA         récupère drapeau chaine et code condition
CF58  LSR A       CFE4  LSR A       et éliminer le drapeau chaine
CF59  STA 2D      CFE5  STA 2D      sauve code de (=)
CF5B  PLA         CFE7  PLA         récupérer opérande de gauche dans ACC2
CF5C  STA D8      CFE8  STA D8      Exposant
CF5E  PLA         CFEA  PLA
CF5F  STA D9      CFEB  STA D9      Octet 1
CF61  PLA         CFED  PLA
CF62  STA DA      CFEE  STA DA      Octet 2
CF64  PLA         CFF0  PLA
CF65  STA DB      CFF1  STA DB      Octet 3 (ou descripteur poids faible)
CF67  PLA         CFF3  PLA
CF68  STA DC      CFF4  STA DC      Octet 4 (ou descripteur poids fort)
CF6A  PLA         CFF6  PLA
CF6B  STA DD      CFF7  STA DD      et signe
CF6D  EOR D5      CFF9  EOR D5      et ajuster le produit des signes
CF6F  STA DE      CFFB  STA DE
CF71  LDA D0      CFFD  LDA D0      sortir avec Z=1 si opérande nul
CF73  RTS         CFFF  RTS         exécuter l'opérateur...ou finir
```

CHERCHER UNE VALEUR DANS ACC1

Entrée: TXTPTR-1 pointe sur le nombre (ou la chaine) à évaluer.

Sortie: l'opérande est dans ACC1, l'évaluation s'étant arrêtée sur le premier caractère incompris, ou bien entendu la fin d'un instruction.

```
CF74  LDA #00     D000  LDA #00     au début,
CF76  STA 28      D002  STA 28      type numérique
CF78  JSR $00E2   D004  JSR $00E2   prendre caractère courant
CF7B  BCS CF80    D007  BCS D00C    sauter si pas chiffre
CF7D  JMP $DFCF   D009  JMP $DFE7   évaluer un nombre  et c'est tout
CF80  JSR $D186   D00C  JSR $D216   est-ce une lettre ?
CF83  BCS CFF0    D00F  BCS D07C    oui,prendre valeur de la variable
CF85  CMP #'.'    D011  CMP #'.'    point décimal ?
CF87  BEQ CF7D    D013  BEQ D009    oui évaluer nombre
CF89  CMP #'#'    D015  CMP #'#'    nombre Hexa ?
CF8B  BEQ CF7D    D017  BEQ D009    oui, évaluer nombre
CF8D  CMP #$2D    D019  CMP #$2D    est ce un - ?
```

```
CF8F  BEQ CFE9    D01B  BEQ D075    oui, éxécuter l'opérateur '-'
CF91  CMP #&+     D01D  CMP #&+     est-ce un + ?
CF93  BEQ CF78    D01F  BEQ D004    oui, l'ignorer et continuer la recherche
CF95  CMP #'"'    D021  CMP #'"'    est-ce le début d'une chaine ?
CF97  BNE CFA8    D023  BNE D034    non, sauter
CF99  LDA E9      D025  LDA E9      C=1
CF9B  LDY EA      D027  LDY EA      AY=TXTPTR
CF9D  ADC #00     D029  ADC #00     ajouter 1
CF9F  BCC CFA2    D02B  BCC D02E
CFA1  INY         D02D  INY         propagation au poids fort
CFA2  JSR $D4FA   D02E  JSR $D5B5   évaluer la chaine pointée par AY
CFA5  JMP $D852   D031  JMP $D90D   récuperer TXTPTR et finir
CFA8  CMP #&NOT   D034  CMP #&NOT   est-ce NOT ?
CFAA  BNE CFBF    D036  BNE D04B    non,continuer
CFAC  LDY #18     D038  LDY #18     indexer opérateur 'NOT'
CFAE  BNE CFEB    D03A  BNE D077    et exécuter l'opérateur
```

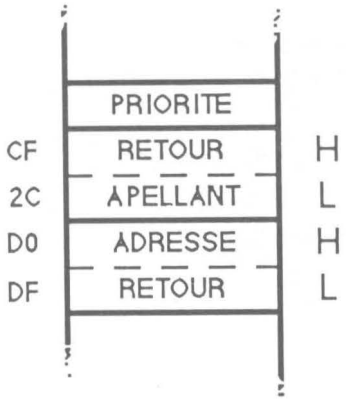
'NOT' (OPERATEUR)

```
CFB0  JSR $D217   D03C  JSR $D2A9   convertir ACC1 en entier dans #D4-#D3
CFB3  LDA D4      D03F  LDA D4      prendre le poids fort
CFB5  EOR #FF     D041  EOR #FF     le complementer
CFB7  TAY         D043  TAY         et le mettre dans Y
CFB8  LDA D3      D044  LDA D3      prendre le poids faible
CFBA  EOR #FF     D046  EOR #FF     le complémenter aussi dans A
CFBC  JMP $D3ED   D048  JMP $D499   YA --> ACC1 (signé)
CFBF  CMP #&FN    D04B  CMP #&FN    est-ce FN ?
CFC1  BNE CFC6    D04D  BNE D052
CFC3  JMP $D467   D04F  JMP $D522   oui, chercher la valeur et sortir
CFC6  CMP #&SGN   D052  CMP #&SGN   tester par rapport à SGN (1ère fonction)
CFC8  BCC CFCD    D054  BCC D059    si pas fonction, sauter
CFCA  JMP $D014   D056  JMP $D0A0   sinon, appliquer la fonction et sortir
CFCD  JSR $CFD6   D059  JSR $D062   demander '('
CFD0  JSR $CE8B   D05C  JSR $CF17   et évaluer l'expression
```

DEMANDER UNE)

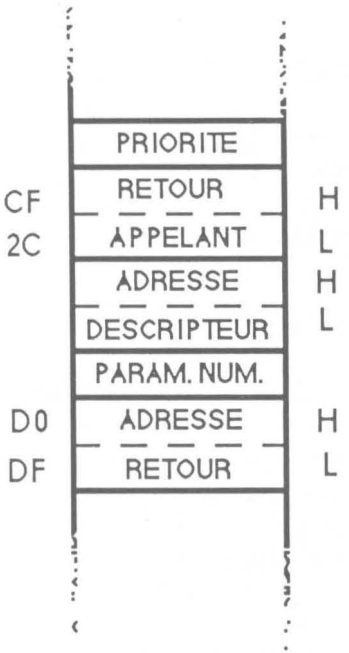
```
CFD3  LDA #')'    D05F  LDA #')'    prendre code à tester
CFD5  BYT #2C     D061  BYT #2C     et sauter l'instruction suivante
```

```
                                DEMANDER UNE (

CFD6  LDA #'('     D062  LDA #'('     prendre le code à tester
CFD8  BYT #2C      D064  BYT #2C      et sauter l'instruction suivante


                                DEMANDER UNE ,

CFD9  LDA #','     D065  LDA #','     prendre le code à tester


                                DEMANDER A

CFDB  LDY #00      D067  LDY #00      préparer Y pour indirection
CFDD  CMP (E9),Y   D069  CMP (E9),Y   et vérifier si bon code
CFDF  BNE CFE4     D06B  BNE D070     non, erreur
CFE1  JMP $00E2    D06D  JMP $00E2    oui, sauter prendre caractère suivant
CFE4  LDX #10      D070  LDX #10      indexer 'SYNTAX'
CFE6  JMP $C485    D072  JMP $C47E    et exécuter l'erreur


                        EXECUTER '-' (changement de signe)

CFE9  LDY #15      D075  LDY #15      indexer opérateur '-'
CFEB  PLA          D077  PLA          enlever l'adresse de retour
CFEC  PLA          D078  PLA
CFED  JMP $CEE7    D079  JMP $CF73    et simuler opérateur


                        PRENDRE LA VALEUR D'UNE VARIABLE

CFF0  JSR $D0FC    D07C  JSR $D188    Prendre adresse de la variable (sans créer)
CFF3  STA D3       D07F  STA D3       et sauver l'adresse
CFF5  STY D4       D081  STY D4       pour indirection
CFF7  LDX 28       D083  LDX 28       X=drapeau numérique/chaine
CFF9  BEQ D000     D085  BEQ D08C     sauter si numérique
CFFB  LDX #00      D087  LDX #00      si chaine, bit d'extension=0
CFFD  STX DF       D089  STX DF
CFFF  RTS          D08B  RTS
D000  LDX 29       D08C  LDX 29       prendre drapeau entier/réel
D002  BPL D011     D08E  BPL D09D     et sauter si réel
D004  LDY #00      D090  LDY #00      préparer indirection
D006  LDA (D3),Y   D092  LDA (D3),Y   prendre poids fort de l'entier
D008  TAX          D094  TAX          dans X
```



Etat de la pile lors de l'appel d'une fonction numérique .

Figure CF-A



Etat de la pile lors de l'évaluation d'une fonction à plusieurs paramètres (MID$ ...)

Figure CF-B

```
D009  INY          D095  INY
D00A  LDA (D3),Y   D096  LDA (D3),Y  et poids faible
D00C  TAY          D098  TAY         dans Y
D00D  TXA          D099  TXA         poids fort dans A à nouveau
D00E  JMP $D3ED    D09A  JMP $D499   YA --> ACC1 (signé)
D011  JMP $DE73    D09D  JMP $DE7B   transfert (AY) -> ACC1
```

EXECUTER UNE FONCTION

```
D014  ASL A        D0A0  ASL A       multiplier par deux le token
D015  PHA          D0A1  PHA         et sauver sur la pile
D016  TAX          D0A2  TAX         et dans X aussi
D017  JSR $00E2    D0A3  JSR $00E2   se placer sur le caractère suivant
D01A  CPX #DB      D0A6  CPX #DB     comparer à code pour CHR$
D01C  BCC D042     D0A8  BCC D0CE    si inférieur ou =,un seul paramétre
D01E  CPX #E7      D0AA  CPX #E7     comparer à code pour KEY$
D020  BCC D045     D0AC  BCC D0D1    si inférieur, aucun paramétre
```

Fonctions à plusieurs paramètres

```
D022  JSR $CFD6    D0AE  JSR $D062   demander '('
D025  JSR $CE8B    D0B1  JSR $CF17   prendre premier paramètre
D028  JSR $CFD9    D0B4  JSR $D065   demander une ','
D02B  JSR $CE7C    D0B7  JSR $CF08   verifier 1er paramètre est une chaine
D02E  PLA          D0BA  PLA         récuperer le token (x2)
D02F  TAX          D0BB  TAX         dans X
D030  LDA D4       D0BC  LDA D4      sauver adresse du descripteur
D032  PHA          D0BE  PHA         sur la pile, poids fort
D033  LDA D3       D0BF  LDA D3
D035  PHA          D0C1  PHA         et poids faible aussi
D036  TXA          D0C2  TXA
D037  PHA          D0C3  PHA         ainsi que le token (x2)
D038  JSR $D00D    D0C4  JSR $D0C8   prendre une valeur entière dans X
D03B  PLA          D0C7  PLA         récuperer token (x2)
D03C  TAY          D0C8  TAY         dans Y pour index
D03D  TXA          D0C9  TXA         mais sauver
D03E  PHA          D0CA  PHA         la valeur du 2ème paramètre
D03F  JMP $D047    D0CB  JMP $D0D3   et exécuter la fonction
```

Fonctions à un seul paramètre

```
D042  JSR $CFCD    D0CE  JSR $D059   demander '(', paramètre, et ')'
```

```
D045  PLA          D0D1  PLA          récupérer le token (x2)
D046  TAY          D0D2  TAY          dans Y pour index
D047  LDA BFDE,Y   D0D3  LDA BFDE,Y   prendre poids faible adresse de la fonction
D04A  STA C4       D0D6  STA C4       et la sauver
D04C  LDA BFDF,Y   D0D8  LDA BFDF,Y   prendre poids fort
D04F  STA C5       D0DB  STA C5       le sauver aussi
D051  JSR $00C3    D0DD  JSR $00C3    exécuter fonction
D054  JMP $CE7A    D0E0  JMP $CF06    et vérifier résultat numérique
```

'OR' (OPERATEUR)

Principe: selon la loi logique non (A ou B)=non A et non B (ou encore A ou B=non (non A et non B)), le OR est traité comme le AND, sauf que les opérateurs puis le résultat sont complémentés.

```
D057  LDY #FF      D0E3  LDY #FF      indiquer complémentation
D059  BYT #2C      D0E5  BYT #2C      et sauter l'instruction suivante
```

'AND' (OPERATEUR)

```
D05A  LDY #$00     D0E6  LDY #$00     indiquer pas de complémentation
D05C  STY 26       D0E8  STY 26       et sauver
D05E  JSR $D217    D0EA  JSR $D2A9    ACC1 --> #D4-#D3 (opérande de droite)
D061  LDA D3       D0ED  LDA D3       prendre poids fort
D063  EOR 26       D0EF  EOR 26       complémenter éventuellement
D065  STA 24       D0F1  STA 24       et sauver
D067  LDA D4       D0F3  LDA D4
D069  EOR 26       D0F5  EOR 26
D06B  STA 25       D0F7  STA 25       idem pour poids faible
D06D  JSR $DECD    D0F9  JSR $DED5    ACC2 --> ACC1: prendre opérande de gauche
D070  JSR $D217    D0FC  JSR $D2A9    ACC1 --> #D4-#D3
D073  LDA D4       D0FF  LDA D4       prendre poids faible
D075  EOR 26       D101  EOR 26       complémenter éventuellement
D077  AND 25       D103  AND 25       et ET avec opérande de gauche
D079  EOR 26       D105  EOR 26       et complémenter éventuellement le résultat
D07B  TAY          D107  TAY          et le sauver dans Y
D07C  LDA D3       D108  LDA D3
D07E  EOR 26       D10A  EOR 26
D080  AND 24       D10C  AND 24
D082  EOR 26       D10E  EOR 26       idem pour le poids fort
D084  JMP $D3ED    D110  JMP $D499    YA --> ACC1 (signé)
```

'<=>' (OPERATEUR)

Principe: c'est une routine très compliquée.Rappelons la structure des variables systémes employées:

    #BC:%0000)=(C
    #2D:%000000)=(

En effet, une seule routine traite tous les codes de comparaisons (=,(),),>= etc...). Ce code est condensé dans ces variables systèmes. De plus, la variable système #BC (d'où est tirée #2D) inclut dans son premier bit un indicateur chaine (1) ou numérique (0).

Comme avec tous les opérateurs, ACC1 contient l'opérande de droite et ACC2 celui de gauche. Rappel aussi, en arrivant C est positionné à 1 (chaine) ou 0 (nombre)

Le résultat classique d'une comparaison (X et Y) est: #00 si X=Y, #01 si Y>X et #FF si X<Y. Dans un premier temps, on va donc essayer de se ramener à ce cas (rappelons que comparer X et Y,c'est évaluer le signe de la différence Y-X).

Mais il est facile de passer de #FF (<), #00 (=), #01 (>) à #00, #01, #02, puisq'il suffit d'ajouter 1. On obtient alors pour le résultat de la comparaison le même motif binaire que le code demandé: %000000)=(. Il suffit ensuite de faire un ET logique entre le résultat obtenu et celui escompté. Si aucune des conditions n'est remplie, cela donne 0. Sinon, cela donne un résultat non nul.

Rappel: >= signifie supérieur OU égal, c'est à dire qu'une seule des deux conditions suffit. Du reste, au sens de la comparaison, > ET = est impossible.

```
D087  JSR $CE7D   D113  JSR $CF09   vérifier type concordant (il est dans C)
D08A  BCS D09F    D116  BCS D12B    sauter si chaine

Comparer deux nombres

D08C  LDA DD      D118  LDA DD      prendre signe ACC2
D08E  ORA #7F     D11A  ORA #7F     et ajouter des 1 pour ne pas
D090  AND D9      D11C  AND D9      toucher à l'exposant...
D092  STA D9      D11E  STA D9      on a maintenant exposant et signe...
D094  LDA #D8     D120  LDA #D8
D096  LDY #00     D122  LDY #00     AY=#00D8=ACC2
D098  JSR $DF34   D124  JSR $DF4C   comparer (AY) et ACC1
D09B  TAX         D127  TAX         résultat dans X
D09C  JMP $D0D2   D128  JMP $D15E   et aller traiter
```

Comparer des chaines

Principe: on va comparer caractère par caractère les deux chaines. Si tous les caractères sont égaux, c'est la longueur qui décidera du résultat. On commence par calculer le résultat de la longueur car il faut de toutes faŁons trouver la plus courte des deux pour effectuer la comparaison.

```
D09F  LDA #00     D12B  LDA #00     indiquer résultat numérique
D0A1  STA 28      D12D  STA 28      et non alphanumérique
D0A3  DEC BC      D12F  DEC BC      enlever drapeau chaine (ramener à 0,1,2)
D0A5  JSR $D715   D131  JSR $D7D0  enlever la réservation (2ème opérande)
D0A8  STA D0      D134  STA D0      sauver longueur d'opérande de droite
D0AA  STX D1      D136  STX D1
D0AC  STY D2      D138  STY D2      et aussi l'adresse de la chaine
D0AE  LDA DB      D13A  LDA DB      prendre adresse descripteur
D0B0  LDY DC      D13C  LDY DC      de l'opérande de gauche
D0B2  JSR $D719   D13E  JSR $D7D4  et enlever sa réservation
D0B5  STX DB      D141  STX DB      sauver adresse de la première chaine
D0B7  STY DC      D143  STY DC
D0B9  TAX         D145  TAX         et longueur dans X
D0BA  SEC         D146  SEC         comparer à longueur de la 2ème chaine
D0BB  SBC D0      D147  SBC D0
D0BD  BEQ D0C7    D149  BEQ D153   si égal, sauter
D0BF  LDA #01     D14B  LDA #01     préparer pour 2ème plus longue
D0C1  BCC D0C7    D14D  BCC D153   oui, c'est bon
D0C3  LDX D0      D14F  LDX D0      la deuxième est plus courte, on prend donc
D0C5  LDA #FF     D151  LDA #FF     sa longueur comme référence et on l'indique
D0C7  STA D5      D153  STA D5      sauver résultat comparaison des longueurs
D0C9  LDY #FF     D155  LDY #FF     préparer index
D0CB  INX         D157  INX         et ajuster longueur de la comparaison
D0CC  INY         D158  INY         caractère suivant
D0CD  DEX         D159  DEX         est-ce le dernier à comparer ?
D0CE  BNE D0D7    D15A  BNE D163   non, sauter
D0D0  LDX D5      D15C  LDX D5      oui, c'est la longueur qui décide
D0D2  BMI D0E3    D15E  BMI D16F
D0D4  CLC         D160  CLC         ajuster C comme la comparaison
D0D5  BCC D0E3    D161  BCC D16F   inconditionnel: effectuer le test
D0D7  LDA (DB),Y  D163  LDA (DB),Y  comparer caractère à caractère
D0D9  CMP (D1),Y  D165  CMP (D1),Y  les deux chaines
D0DB  BEQ D0CC    D167  BEQ D158   si pareil, au suivant
D0DD  LDX #FF     D169  LDX #FF     si différent, calculer le signe
D0DF  BCS D0E3    D16B  BCS D16F   de la comparaison
D0E1  LDX #01     D16D  LDX #01     comme pour les longueurs des chaines
D0E3  INX         D16F  INX         X=0, 1 ou 2
```

```
D0E4  TXA          D170  TXA          dans A
D0E5  ROL A        D171  ROL A        X=1, 2 ou 4: %000000)=(
D0E6  AND 2D       D172  AND 2D       est-ce ce qui est demandé ?
D0E8  BEQ D0EC     D174  BEQ D178     non, FALSE
D0EA  LDA #FF      D176  LDA #FF      oui, TRUE
D0EC  JMP $DF15    D178  JMP $DF24


D0EF  JSR $CFD9    D17B  JSR $D065    demander une ','
```

'DIM' (COMMANDE)

Principe: en fait, la routine suivante s'occupe de tout. Simplement, on met dans X quelque chose de non nul, qui va indiquer que c'est une déclaration de variable, et non une simple consultation.

A est normalement non nul puisqu'il contient le premier caractère de la variable. Si tel n'est pas le cas, une erreur de syntaxe sera de tout façon générée. De même, A étant une lettre, b6=1 (code supérieur à #40. Ces deux caractéristiques (<>0 ou b6=1) seront indifféremment testées.

```
D0F2  TAX          D17E  TAX          X<>0 indique déclaration
D0F3  JSR $D101    D17F  JSR $D18D    et calculer adresse de la variable
D0F6  JSR $00E8    D182  JSR $00E8    prendre caractère courant
D0F9  BNE D0EF     D185  BNE D17B     on recommence une autre déclaration
D0FB  RTS          D187  RTS
```

PRENDRE ADRESSE D'UNE VARIABLE

Entrée: TXTPTR pointe sur la variable à trouver et #2B contient un code permettant d'empêcher éventuellement certains types. #2B est remis à 0 lors d'un CLEAR explicite ou implicite.

#2B=#00: tous types autorisés (entiers, chaines, tableaux)

#2B=#40: on veut seulement un tableau, et seulement son nom, tel que c'est précisé pour un STORE ou RECALL par exemple. (inopérant sur V1.0)

#2B=#80: on interdit les tableaux et les entiers (par exemple pour les indices des boucles FOR-NEXT)

#27 indique un DIM (non nul, b6=1)

Sortie: AY pointe sur l'adresse de la variable, là où doit être placée sa valeur. On a aussi #B6-#B7=AY

Principe: c'est la routine centrale de la gestion des variables. On commence

par saisir son nom (y compris son type), en regardant évidemment s'il est autorisé. Si c'est une variable, et pas un tableau, on la recherche, et la crée éventuellement, c'est donc assez simple.

Pour les tableaux, on saisi d'abord son nom, puis on stocke sa dimension et les composantes dans chaque dimension. Ensuite, on recherche le tableau. Si on le trouve, on vérifie qu'on ne vient pas de DIM (erreur), et on calcule l'adresse de l'élément.

Si le tableau n'est pas trouvé, on va sortir s'il s'agit d'une simple consultation. Sinon, le tableau va être créé. Chaque composante aura alors la valeur précisée (si on vient de DIM) ou 10 par défaut s'il s'agit d'un dimensionnement implicite. On calculera enfin l'adresse de l'élément.

```
D0FC  LDX #00      D188  LDX #00      indiquer consultation (pour les tableaux)
D0FE  JSR $00E8    D18A  JSR $00E8    prendre premier caractère de la variable
D101  STX 27       D18D  STX 27       sauver le drapeau consultation/déclaration
D103  STA B4       D18F  STA B4       et le premier caractère de la variable
D105  JSR $00E8    D191  JSR $00E8    prendre 1er caractère (si DIM)
D108  JSR $D186    D194  JSR $D216    et vérifier alphabétique
D10B  BCS D110     D197  BCS D19C
D10D  JMP $CFE4    D199  JMP $D070    non,'SYNTAX ERROR'
D110  LDX #00      D19C  LDX #00      2ème caractère significatif=0
D112  STX 28       D19E  STX 28       et initialiser drapeau chaine
D114  STX 29       D1A0  STX 29       et aussi drapeau entier
D116  JSR $00E2    D1A2  JSR $00E2    prendre caractère suivant
D119  BCC D120     D1A5  BCC D1AC     si numérique,c'est bon
D11B  JSR $D186    D1A7  JSR $D216    est-ce alphabétique ?
D11E  BCC D12B     D1AA  BCC D1B7     non,on saute
D120  TAX          D1AC  TAX          oui,dans X le 2ème caractère
D121  JSR $00E2    D1AD  JSR $00E2    et on saute le 2ème caractère
D124  BCC D121     D1B0  BCC D1AD     et aussi tous ceux non significatifs
D126  JSR $D186    D1B2  JSR $D216    en vérifiant qu'ils sont bien numériques
D129  BCS D121     D1B5  BCS D1AD     ou alphabétiques
D12B  CMP #'$'     D1B7  CMP #'$'     est-ce un indicateur de chaine ?
D12D  BNE D135     D1B9  BNE D1C1     non,on saute
D12F  LDA #FF      D1BB  LDA #FF      oui,on place drapeau chaine
D131  STA 28       D1BD  STA 28
D133  BNE D145     D1BF  BNE D1D1     inconditionnel: on ajuste la variable
D135  CMP #'%'     D1C1  CMP #'%'     est-ce un indicateur d'entier ?
D137  BNE D14C     D1C3  BNE D1D8     non, aller ajuster la variable
D139  LDA 2B       D1C5  LDA 2B       oui, au fait entier permis ?
D13B  BNE D10D     D1C7  BMI D199     non, SYNTAX ERROR
D13D  LDA #80      D1C9  LDA #80      oui, placer drapeau entier
D13F  STA 29       D1CB  STA 29
D141  ORA B4       D1CD  ORA B4       et ajuster tout de suite
```

```
D143  STA B4      D1CF  STA B4      le nom de la variable
D145  TXA         D1D1  TXA         X contient toujours le 2ème caractère
D146  ORA #80     D1D2  ORA #80     on indique entier ou chaine
D148  TAX         D1D4  TAX         selon d'où on vient
D149  JSR $00E2   D1D5  JSR $00E2   on saute l'indicateur de type
D14C  STX B5      D1D8  STX B5      et on sauve enfin le 2ème caractère
D14E  SEC         D1DA  SEC
D14F  ORA 2B      D1DB  ORA 2B      on ajoute le drapeau d'autorisation
D151  SBC #'('    D1DD  SBC #'('    et on enlève code de '('
D153  BNE D158    D1DF  BNE D1E4    si '(' et matrice autorisée,
D155  JMP $D229   D1E1  JMP $D2BB   on va traiter le tableau
.....             D1E4  BIT 2B      sinon, voulait-on simplement le nom ?
.....             D1E6  BVS D1E1    (STORE notamment) oui, traiter le tableau
```

Chercher une variable

```
D158  LDA #00     D1E8  LDA #00     réinitialiser le drapeau d'autorisation
D15A  STA 2B      D1EA  STA 2B
D15C  LDA 9C      D1EC  LDA 9C
D15E  LDX 9D      D1EE  LDX 9D      AX=début des variables
D160  LDY #00     D1F0  LDY #00     et préparer l'index
D162  STX CF      D1F2  STX CF      sauver adresse dans pointeur de travail
D164  STA CE      D1F4  STA CE
D166  CPX 9F      D1F6  CPX 9F      fin de la zone des variables atteinte ?
D168  BNE D16E    D1F8  BNE D1FE    non, on continue
D16A  CMP 9E      D1FA  CMP 9E      on compare aussi le poids faible
D16C  BEQ D190    D1FC  BEQ D222    aller traiter variable non trouvée
D16E  LDA B4      D1FE  LDA B4      prendre premier caractère variable
D170  CMP (CE),Y  D200  CMP (CE),Y  et comparer
D172  BNE D17C    D202  BNE D20C    déjà pas bon, on passe à la suivante
D174  LDA B5      D204  LDA B5      on prend le deuxième
D176  INY         D206  INY         on ajuste l'index
D177  CMP (CE),Y  D207  CMP (CE),Y  et on compare
D179  BEQ D1E5    D209  BEQ D277    sortir avec variable trouvée
D17B  DEY         D20B  DEY         Y=0
D17C  CLC         D20C  CLC         et passer à la variable suivante
D17D  LDA CE      D20D  LDA CE      c'est à dire ajouter 7
D17F  ADC #07     D20F  ADC #07     au pointeur courant
D181  BCC D164    D211  BCC D1F4   sans oublier le poids fort
D183  INX         D213  INX         qui est toujours dans X
D184  BNE D162    D214  BNE D1F2    et recommencer
```

TESTER SI A EST ALPHABETIQUE

Entrée: A contient le code du caractère à tester

Sortie: C=Ø si pas alphabétique, C=1 sinon. Les minuscules ne sont pas alphabétiques.
A,X,Y inchangés.

Principe: une astuce est utilisée pour placer C correctement: il faut en fait inverser sa signification, puisqu'on doit trouver C=1 pour un code inférieur à 'I', alors qu'une comparaison normale donne l'inverse.

La première soustraction va donner un nombre supérieur à #A5 (entre #E6 et #FF) si c'est un caractère alphabétique, de sorte que la soustraction de #A5 se fera normalement, et on sort donc avec C=1.

Si le caractère n'est pas alphabétique, la première soustraction donnera un résultat inférieur à #A5, de sorte que la 2ème soustraction créera un report (C=Ø).

Bien entendu, #A5+#5B=#1ØØ, soit en fait Ø, ce qui permet de retrouver A.

```
D186  CMP #'A'     D216  CMP #'A'     comparer à première lettre alphabet
D188  BCC D18F     D218  BCC D221     au dessous, on sort, C=Ø
D18A  SBC #' '     D21A  SBC #' '     C=1 on enlève le code du premier après Z
D18C  SEC          D21C  SEC
D18D  SBC #A5      D21D  SBC #A5      et on réajuste A, tout en plaŁant C
..............     D21F  BCS D221     n'importe quoi !
D18F  RTS          D221  RTS
```

Traiter une variable non trouvée

Principe: dans le cas ou la routine est appelée par une simple consultation, il est inutile de créer la variable, on retourne simplement une adresse qui correspond à une valeur Ø (exposant nul) où à la chaine vide (longueur nulle).

L'adresse d'appel (ou adresse de retour...) se trouve sur la pile, il suffit donc de la tester.

```
D19Ø  PLA          D222  PLA          récupérer adresse d'appel,poids faible
D191  PHA          D223  PHA          et réajuster la pile
D192  CMP #F2      D224  CMP #7E      et tester pour #CFF2/#DØ7E
D194  BNE D1A3     D226  BNE D235     soit en fait le JSR placé en #CFFØ/#DØ7C
D196  TSX          D228  TSX          si poids faible OK,
D197  LDA Ø1Ø2,X   D229  LDA Ø1Ø2,X   tester poids fort aussi
D19A  CMP #CF      D22C  CMP #DØ
D19C  BNE D1A3     D22E  BNE D235     pas bon, on va créer la variable
```

```
D19E  LDA #03     D230  LDA #07     appel par consultation, AY pointe
D1A0  LDY #E2     D232  LDY #E2     sur une valeur bidon (#00 #00 #00,soit nul
D1A2  RTS         D234  RTS         si numérique ou chaine vide )
```

Créer une variable

Principe: assez simple, on décale vers le haut les tableaux, de la longueur d'une variable et on insère la variable (7 octets).

NB: #A0-#A1 est automatiquement placé par #C3F8/#C3F4)

On finit comme si la variable avait été trouvée, puisqu'on est en fait dans le même cas, après la création.

```
D1A3  LDA 9E      D235  LDA 9E
D1A5  LDY 9F      D237  LDY 9F      prendre fin des variables
D1A7  STA CE      D239  STA CE
D1A9  STY CF      D23B  STY CF      comme début de la zone à décaler
D1AB  LDA A0      D23D  LDA A0
D1AD  LDY A1      D23F  LDY A1      prendre haut des tableaux
D1AF  STA C9      D241  STA C9
D1B1  STY CA      D243  STY CA      comme fin de la zone à décaler
D1B3  CLC         D245  CLC
D1B4  ADC #07     D246  ADC #07     il faut 7 octets pour une variable
D1B6  BCC D1B9    D248  BCC D24B    on calcule l'adresse cible
D1B8  INY         D24A  INY         sans oublier le poids fort...
D1B9  STA C7      D24B  STA C7
D1BB  STY C8      D24D  STY C8      dans AY aussi...
D1BD  JSR $C3F8   D24F  JSR $C3F4   on effectue le décalage
D1C0  LDA C7      D252  LDA C7
D1C2  LDY C8      D254  LDY C8      on récupère le nouveau bas des tableaux
D1C4  INY         D256  INY         en ajustant le poids fort
D1C5  STA 9E      D257  STA 9E      (c'est plus court qu'ajouter 7)
D1C7  STY 9F      D259  STY 9F      et on l'initialise.
D1C9  LDY #00     D25B  LDY #00     préparer l'index
D1CB  LDA B4      D25D  LDA B4      prendre premier caractère du nom
D1CD  STA (CE),Y  D25F  STA (CE),Y  et le placer dans la variable
D1CF  INY         D261  INY
D1D0  LDA B5      D262  LDA B5      idem pour le deuxième caractère du nom
D1D2  STA (CE),Y  D264  STA (CE),Y  dans la variable aussi
D1D4  LDA #00     D266  LDA #00     et annuler la zone résultat
D1D6  INY         D268  INY
D1D7  STA (CE),Y  D269  STA (CE),Y  une boucle aurait été plus courte
D1D9  INY         D26B  INY
D1DA  STA (CE),Y  D26C  STA (CE),Y  surtout que la rapidité
D1DC  INY         D26E  INY
```

```
D1DD  STA (CE),Y  D26F  STA (CE),Y  n'est pas critique ici
D1DF  INY         D271  INY
D1E0  STA (CE),Y  D272  STA (CE),Y
D1E2  INY         D274  INY
D1E3  STA (CE),Y  D275  STA (CE),Y
Variable trouvée

D1E5  LDA CE      D277  LDA CE      prendre adresse de la variable
D1E7  CLC         D279  CLC         (pointant sur son nom)
D1E8  ADC #02     D27A  ADC #02     et l'ajuster sur le premier
D1EA  LDY CF      D27C  LDY CF      octet de sa valeur
D1EC  BCC D1EF    D27E  BCC D281
D1EE  INY         D280  INY         sans oublier le poids fort.
D1EF  STA B6      D281  STA B6      la sauver en #B6-#B7
D1F1  STY B7      D283  STY B7
D1F3  RTS         D285  RTS

Calculer adresse du début effectif de la matrice

D1F4  LDA 26      D286  LDA 26      prendre nombre de dimensions
D1F6  ASL A       D288  ASL A       x2
D1F7  ADC #05     D289  ADC #05     +5 (2:nom,2:longueur,1:nombre de dimension)
D1F9  ADC CE      D28B  ADC CE      et l'ajouter au début
D1FB  LDY CF      D28D  LDY CF      du tableau
D1FD  BCC D200    D28F  BCC D292
D1FF  INY         D291  INY         sans oublier le poids fort
D200  STA C7      D292  STA C7      on a maintenant l'adresse dans AY
D202  STY C8      D294  STY C8      et dans #C7-#C8
D204  RTS         D296  RTS

D205 D297  BYT #90,#80,#00,#00,#00  soit -32768 ou encore -#8000

                        PRENDRE UN ENTIER NON SIGNE

Entrée: TXTPTR-1 bien placé

Sortie: un entier non signé dans #33-#34, #D4-#D3, YA.

D20A  JSR $00E2   D29C  JSR $00E2   sauter un caractère
D20D  JSR $CE8B   D29F  JSR $CF17   évaluer l'expression
D210  JSR $CE7A   D2A2  JSR $CF06   vérifier numérique (et #CE77/#CF03 ?)
D213  LDA D5      D2A5  LDA D5      prendre signe
D215  BMI D224    D2A7  BMI D2B6    si négatif, erreur
```

ACC1 --> #D4-#D3 (non signé)

Entrée: un nombre dans ACC1.

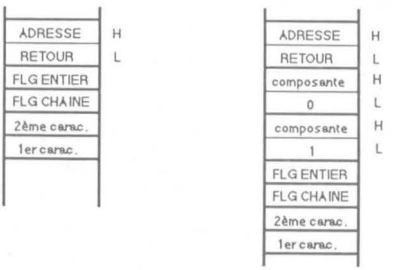
Sortie: cf ci-dessus.

```
D217  LDA D0      D2A9  LDA D0      prendre exposant
D219  CMP #90     D2AB  CMP #90     comparer pour #8000
D21B  BCC D226    D2AD  BCC D2B8    inférieur,c'est bon
D21D  LDA #05     D2AF  LDA #97
D21F  LDY #D2     D2B1  LDY #D2     AY pointe sur -#8000
D221  JSR $DF34   D2B3  JSR $DF4C   comparer (AY) et ACC1
D224  BNE D2A0    D2B6  BNE D336    'ILLEGAL QUANTITY ERROR' si < ou =
D226  JMP $DF74   D2B8  JMP $DF8C   ACC1 --> entier (#D4-#D3-#D2-#D1)
```

Traiter un tableau

Principe: on se débrouille pour avoir toujours au sommet de la pile le nom de la variable, son type, et le nombre de dimensions, tout en empilant les composantes de chaque dimensions. On échange constamment les drapeaux et les composantes,qui font tous deux 2 octets. Il est aussi nécéssaire de sauver le nom que le type du tableau, car l'évaluation des composantes modifient ces variables systèmes.

FIGURE D0-A



Etat de la pile lors de l'évaluation des composantes d'un tableau .

```
.............  D2BB  LDA 2B      prendre drapeau autorisation
.............  D2BD  BNE D306    si non nul (#40),simplement adresse début.
D229  LDA 27      D2BF  LDA 27      drapeau consultation
D22B  ORA 29      D2C1  ORA 29      en même temps que b7 dit si entier
D22D  PHA         D2C3  PHA         et on sauve
D22E  LDA 28      D2C4  LDA 28      drapeau chaine
D230  PHA         D2C6  PHA         on le sauve aussi
D231  LDY #00     D2C7  LDY #00     initialiser le compteur de dimensions
D233  TYA         D2C9  TYA
D234  PHA         D2CA  PHA         sauver la dimension courante
D235  LDA B5      D2CB  LDA B5
D237  PHA         D2CD  PHA         et aussi nom tableau
D238  LDA B4      D2CE  LDA B4
D23A  PHA         D2D0  PHA         les 2 caractères
```

```
D23B  JSR $D20A   D2D1  JSR $D29C   prendre composante de la dimension courante
D23E  PLA         D2D4  PLA
D23F  STA B4      D2D5  STA B4      et récupérer le nom de la variable
D241  PLA         D2D7  PLA
D242  STA B5      D2D8  STA B5
D244  PLA         D2DA  PLA
D245  TAY         D2DB  TAY         et le numéro de la dimension courante
D246  TSX         D2DC  TSX
D247  LDA 0102,X  D2DD  LDA 0102,X  on prend le drapeau entier/consultation
D24A  PHA         D2E0  PHA         et on le remet plus accessible
D24B  LDA 0101,X  D2E1  LDA 0101,X  idem pour le drapeau chaine
D24E  PHA         D2E4  PHA         qui est maintenant au sommet
D24F  LDA D3      D2E5  LDA D3      on remplace par la composante
D251  STA 0102,X  D2E7  STA 0102,X  poids fort
D254  LDA D4      D2EA  LDA D4
D256  STA 0101,X  D2EC  STA 0101,X  puis poids faible
D259  INY         D2EF  INY         on prépare la dimension suivante
D25A  JSR $00E8   D2F0  JSR $00E8   et on prend le caractère courant
D25D  CMP #','    D2F3  CMP #','    une autre dimension ?
D25F  BEQ D233    D2F5  BEQ D2C9    oui,on recommence
D261  STY 26      D2F7  STY 26      non,on sauve le nombre de dimension
D263  JSR $CFD3   D2F9  JSR $D05F   et on demande une ')'
D266  PLA         D2FC  PLA         on récupère le drapeau chaine
D267  STA 28      D2FD  STA 28
D269  PLA         D2FF  PLA         et le drapeau mixte entier/consultation
D26A  STA 29      D300  STA 29      seul b7 importe pour le drapeau entier
D26C  AND #7F     D302  AND #7F     en revanche b7 ne doit pas influencer
D26E  STA 27      D304  STA 27      le drapeau consultation
```

Trouver le tableau

Principe: aussi simple que pour une variable simple, mais on saute d'un tableau à l'autre grâce à la longueur du tableau.

```
D270  LDX 9E      D306  LDX 9E
D272  LDA 9F      D308  LDA 9F      prendre début de la zone des tableaux
D274  STX CE      D30A  STX CE
D276  STA CF      D30C  STA CF      et sauver pour pointeur
D278  CMP A1      D30E  CMP A1      comparer à la fin des tableaux
D27A  BNE D280    D310  BNE D316    pas encore atteint,on continue
D27C  CPX A0      D312  CPX A0      on compare aussi le poids faible
D27E  BEQ D2B9    D314  BEQ D355    c'est la fin et on n'a pas trouvé:on crée
D280  LDY #00     D316  LDY #00     préparer l'index
D282  LDA (CE),Y  D318  LDA (CE),Y  premier caractère
```

```
D284  INY          D31A  INY
D285  CMP B4       D31B  CMP B4       c'est déjà pas bon,on passe au suivant
D287  BNE D28F     D31D  BNE D325
D289  LDA B5       D31F  LDA B5       deuxième caractère
D28B  CMP (CE),Y   D321  CMP (CE),Y
D28D  BEQ D2A5     D323  BEQ D33B     on l'a trouvé !
D28F  INY          D325  INY          passer au suivant:on index la longueur
D290  LDA (CE),Y   D326  LDA (CE),Y
D292  CLC          D328  CLC
D293  ADC CE       D329  ADC CE       et on l'ajouter au pointeur courant
D295  TAX          D32B  TAX          poids faible dans X
D296  INY          D32C  INY
D297  LDA (CE),Y   D32D  LDA (CE),Y   et idem pour le poids fort
D299  ADC CF       D32F  ADC CF
D29B  BCC D274     D331  BCC D30A     inconditionnel:on continue la recherche...
D29D  LDX #6B      D333  LDX #6B      'BAD SUBSCRIPIT ERROR'
D29F  BYT #2C      D335  BYT #2C
D2A0  LDX #35      D336  LDX #35      'ILLEGAL QUANTITY ERROR'
D2A2  JMP $C485    D338  JMP $C47E
```

Le tableau est trouvé

```
D2A5  LDX #78      D33B  LDX #78      préparer 'REDIM'D ARRAY ERROR'
D2A7  LDA 27       D33D  LDA 27       drapeau consultation
D2A9  BNE D2A2     D33F  BNE D338     erreur si déclaration
.....              D341  LDA 2B      drapeau autorisation
.....              D343  BEQ D347     sauter si OK
.....              D345  SEC          si on voulait juste l'adresse,on sort
.....              D346  RTS          C=1 indique qu'on l'a trouvé
D2AB  JSR $D1F4    D347  JSR $D286     calculer l'adresse effective du tableau
D2AE  LDA 26       D34A  LDA 26       prendre la nombre de dimension
D2B0  LDY #04      D34C  LDY #04
D2B2  CMP (CE),Y   D34E  CMP (CE),Y   et le comparer à valeur réelle
D2B4  BNE D29D     D350  BNE D333     si pas pareil, erreur !
D2B6  JMP $D343    D352  JMP $D3EB    si OK,calculer l'adresse de l'élément.
```

CREER UNE MATRICE

```
.....              D355  LDA 2B       prendre drapeau d'autorisation
.....              D357  BEQ D361     OK,on continue
```

Sortie de RECALL:Tableau non dimensionné

```
.............. D359 JSR $E93D  reconfigurer le VIA
.............. D35C LDX #2A    'OUT OF DATA ERROR'
.............. D35E JMP $C47E

D2B9 JSR $D1F4   D361 JSR $D286  calcul du début effectif de la matrice
D2BC JSR $C448   D364 JSR $C444  et vérifier qu'il y a la place
D2BF LDA #00     D367 LDA #00    initialiser longueur poids faible
D2C1 TAY         D369 TAY        préparer index
D2C2 STA E1      D36A STA E1     et longueur poids fort à 0 pour l'instant
D2C4 LDX #05     D36C LDX #05    initialiser longueur d'un élement réel
D2C6 LDA B4      D36E LDA B4     premier caractère du nom
D2C8 STA (CE),Y  D370 STA (CE),Y et le sauver
D2CA BPL D2CD    D372 BPL D375   si réel,conserver la longueur
D2CC DEX         D374 DEX        longueur élément=4 pour l'instant si entier
D2CD INY         D375 INY
D2CE LDA B5      D376 LDA B5     deuxième caractère du nom
D2D0 STA (CE),Y  D378 STA (CE),Y et le sauver
D2D2 BPL D2D6    D37A BPL D37E   si réel,conserver la longueur
D2D4 DEX         D37C DEX
D2D5 DEX         D37D DEX        si pas réel,3 pour chaine et 2 pour entier
D2D6 STX E0      D37E STX E0     sauver la longueur poids faible
D2D8 LDA 26      D380 LDA 26     prendre nombre de dimensions
D2DA INY         D382 INY        ajuster Y
D2DB INY         D383 INY
D2DC INY         D384 INY        sur le nombre de dimensions
D2DD STA (CE),Y  D385 STA (CE),Y et le sauver
D2DF LDX #0B     D387 LDX #0B    initialiser composante par défaut (11)
D2E1 LDA #00     D389 LDA #00    poids fort aussi
D2E3 BIT 27      D38B BIT 27     déclaration implicite ou DIM ?
D2E5 BVC D2EF    D38D BVC D397   sauter si implicite (on garde 11)
D2E7 PLA         D38F PLA        récupérer composante poids faible
D2E8 CLC         D390 CLC        et ajuster (élément No 0)
D2E9 ADC #01     D391 ADC #01
D2EB TAX         D393 TAX        et préciser poids faible
D2EC PLA         D394 PLA        prendre poids fort
D2ED ADC #00     D395 ADC #00    ajuster éventuellement
D2EF INY         D397 INY
D2F0 STA (CE),Y  D398 STA (CE),Y et placer poids fort
D2F2 INY         D39A INY
D2F3 TXA         D39B TXA
D2F4 STA (CE),Y  D39C STA (CE),Y puis poids faible (Attention !)
D2F6 JSR $D3A5   D39E JSR $D44D  calculer la longueur courante du tableau
```

```
D2F9  STX E0      D3A1  STX E0      et la sauver
D2FB  STA E1      D3A3  STA E1
D2FD  LDY 91      D3A5  LDY 91      récupérer Y (sauvé par #D3A5/#D44D)
D2FF  DEC 26      D3A7  DEC 26      passer à la dimension suivante
D301  BNE D2DF    D3A9  BNE D387    s'il en reste...
D303  ADC C8      D3AB  ADC C8      ajuster l'adresse de fin (C=0)
D305  BCS D364    D3AD  BCS D40C    si dépassement,'OUT OF MEMORY ERROR'
D307  STA C8      D3AF  STA C8      on sauve le poids fort
D309  TAY         D3B1  TAY         et on sauve dans Y le poids fort
D30A  TXA         D3B2  TXA         prendre poids faible
D30B  ADC C7      D3B3  ADC C7      on ajoute aussi
D30D  BCC D312    D3B5  BCC D3BA     et on n'oublie pas le report éventuel
D30F  INY         D3B7  INY
D310  BEQ D364    D3B8  BEQ D40C    si dépassement, erreur
D312  JSR $C448   D3BA  JSR $C444   y-a-t-il la place ?
D315  STA A0      D3BD  STA A0
D317  STY A1      D3BF  STY A1      oui,on ajuste le haut des tableaux
D319  LDA #00     D3C1  LDA #00     préparer pour remplir de 00
D31B  INC E1      D3C3  INC E1      ajuster longueur zone à effacer (DEC:BNE)
D31D  LDY E0      D3C5  LDY E0      poids faible de la longueur
D31F  BEQ D326    D3C7  BEQ D3CE    si 0,pas de morceau de page
D321  DEY         D3C9  DEY
D322  STA (C7),Y  D3CA  STA (C7),Y
D324  BNE D321    D3CC  BNE D3C9    vider jusqu'à la fin de la page
D326  DEC C8      D3CE  DEC C8      page suivante pointeur
D328  DEC E1      D3D0  DEC E1      et aussi compteur
D32A  BNE D321    D3D2  BNE D3C9    il en reste,on continue
D32C  INC C8      D3D4  INC C8      on réajuste car décrémenté une fois de trop
D32E  SEC         D3D6  SEC
D32F  LDA A0      D3D7  LDA A0      fin du tableau
D331  SBC CE      D3D9  SBC CE      -début=Longueur du tableau
D333  LDY #02     D3DB  LDY #02
D335  STA (CE),Y  D3DD  STA (CE),Y  et la sauver
D337  LDA A1      D3DF  LDA A1      et on n'oublie pas le poids fort
D339  INY         D3E1  INY
D33A  SBC CF      D3E2  SBC CF
D33C  STA (CE),Y  D3E4  STA (CE),Y  que l'on sauve aussi
D33E  LDA 27      D3E6  LDA 27      tester si DIM
D340  BNE D3A4    D3E8  BNE D44C    oui,c'est fini
D342  INY         D3EA  INY         non,on indexe le nombre d'élément
```

Calculer l'adresse d'un élément

Principe: voir le codage d'une matrice (chapitre IV B) 2-).

```
D343  LDA (CE),Y  D3EB  LDA (CE),Y  prendre nombre de dimensions
D345  STA 26      D3ED  STA 26      et le sauver
D347  LDA #ØØ     D3EF  LDA #ØØ
D349  STA EØ      D3F1  STA EØ      initialiser résultat
D34B  STA E1      D3F3  STA E1      à Ø pour l'instant.
D34D  INY         D3F5  INY         indexer la composante
D34E  PLA         D3F6  PLA
D34F  TAX         D3F7  TAX         récupérer composante sur la pile
D35Ø  STA D3      D3F8  STA D3      et sauver (poids faible)
D352  PLA         D3FA  PLA
D353  STA D4      D3FB  STA D4      et idem poids fort
D355  CMP (CE),Y  D3FD  CMP (CE),Y  comparer à maxi permis par DIM
D357  BCC D367    D3FF  BCC D40F    si plus petit,c'est bon
D359  BNE D361    D401  BNE D409    'BAD SUBSCRIPIT'
D35B  INY         D403  INY         ajuster index
D35C  TXA         D404  TXA         reprendre poids faible
D35D  CMP (CE),Y  D405  CMP (CE),Y  et comparer aussi
D35F  BCC D368    D407  BCC D41Ø    si plus petit,c'est bon
D361  JMP $D29D   D409  JMP $D333   'BAD SUBSCRIPIT'
D364  JMP $C483   D40C  JMP $C47C   'OUT OF MEMORY'
D367  INY         D40F  INY
D368  LDA E1      D41Ø  LDA E1      prendre le numéro d'élément
D36A  ORA EØ      D412  ORA EØ      c'est Ø pour l'instant ?
D36C  CLC         D414  CLC         ajuster C si saut
D36D  BEQ D379    D415  BEQ D421   et sauter la multiplication (ØxX=Ø)
D36F  JSR $D3A5   D417  JSR $D44D   calculer numéro de l'élément dans XY
D372  TXA         D41A  TXA         et ajouter valeur de la composante
D373  ADC D3      D41B  ADC D3      poids faible
D375  TAX         D41D  TAX         et on sauve toujours dans X
D376  TYA         D41E  TYA         idem pour le poids fort de la composante
D377  LDY 91      D41F  LDY 91      on récupère l'index
D379  ADC D4      D421  ADC D4
D37B  STX EØ      D423  STX EØ      on sauve le poids faible
D37D  DEC 26      D425  DEC 26      a-t-on pris en compte toutes composantes ?
D37F  BNE D34B    D427  BNE D3F3   non,on continue
D381  STA E1      D429  STA E1      sauver poids fort du numéro de l'élément
D383  LDX #Ø5     D42B  LDX #Ø5     X=Longueur d'un élément réel
D385  LDA B4      D42D  LDA B4
D387  BPL D38A    D42F  BPL D432    conserver si réel
D389  DEX         D431  DEX         X=4 si entier
D38A  LDA B5      D432  LDA B5
D38C  BPL D39Ø    D434  BPL D438    conserver si réel
D38E  DEX         D436  DEX
D38F  DEX         D437  DEX         2 pour entier,3 pour chaine
```

```
D39Ø  STX 97      D438  STX 97      et sauver
D392  LDA #ØØ     D43A  LDA #ØØ     indiquer poids fort=Ø
D394  JSR $D3AE   D43C  JSR $D456   et calculer numéro x longueur
D397  TXA         D43F  TXA
D398  ADC C7      D44Ø  ADC C7      ajouter à l'adresse de début du tableau
D39A  STA B6      D442  STA B6      et sauver
D39C  TYA         D444  TYA
D39D  ADC C8      D445  ADC C8      idem poids fort
D39F  STA B7      D447  STA B7      sauver aussi
D3A1  TAY         D449  TAY         et mettre l'adresse
D3A2  LDA B6      D44A  LDA B6      dans AY et #B6-#B7
D3A4  RTS         D44C  RTS
```

Calculer dans XA et XY #EØ-#E1xcomposante

Entrée: #CE pointe sur le début du tableau, Y sur la composante de la dimension, et #EØ-E1 contient soit le numéro de l'élément (si on vient du calcul d'adresse) soit le dépacement dans le tableau (si on vient de la création).
Sortie: Y en #91, XA=XY=#EØ-#E1 x composante on a aussi C=Ø.

```
D3A5  STY 91      D44D  STY 91      sauver l'index
D3A7  LDA (CE),Y  D44F  LDA (CE),Y  prendre composante poids faible
D3A9  STA 97      D451  STA 97      et la sauver
D3AB  DEY         D453  DEY
D3AC  LDA (CE),Y  D454  LDA (CE),Y  prendre composante poids fort
D3AE  STA 98      D456  STA 98      et sauver aussi
D3BØ  LDA #1Ø     D458  LDA #1Ø     il y aura 16 décalages
D3B2  STA CC      D45A  STA CC      et sauver compteur
D3B4  LDX #ØØ     D45C  LDX #ØØ     résultat poids faible =Ø
D3B6  LDY #ØØ     D45E  LDY #ØØ     résultat poids fort aussi
D3B8  TXA         D46Ø  TXA
D3B9  ASL A       D461  ASL A       décaler le résultat
D3BA  TAX         D462  TAX
D3BB  TYA         D463  TYA
D3BC  ROL A       D464  ROL A       et aussi poids fort
D3BD  TAY         D465  TAY
D3BE  BCS D364    D466  BCS D40C    c'est trop long:'OUT OF MEMORY ERROR'
D3CØ  ASL EØ      D468  ASL EØ
D3C2  ROL E1      D46A  ROL E1      décaler multiplicande
D3C4  BCC D3D1    D46C  BCC D479
D3C6  CLC         D46E  CLC         si il sort 1,
D3C7  TXA         D46F  TXA         on ajoute la composante
D3C8  ADC 97      D47Ø  ADC 97      poids faible
D3CA  TAX         D472  TAX         et on sauve le résultat
```

```
D3CB  TYA          D473  TYA          idem pour le poids faible
D3CC  ADC 98       D474  ADC 98
D3CE  TAY          D476  TAY
D3CF  BCS D364     D477  BCS D40C     dépassement: erreur
D3D1  DEC CC       D479  DEC CC       encore un décalage ?
D3D3  BNE D3B8     D47B  BNE D460     oui,on continue.
D3D5  RTS          D47D  RTS
```

'FRE' (FONCTION)

Principe: on réorganise les chaines pourqu'elles occupent un espace minimum, et on calcule l'espace laissé libre. L'argument peut être indifféremment numérique ou chaine, il ne sert de toutes façons à rien.

Bogue: sur la V1.0, le type du résultat n'est pas forcé à numérique. Donc, A=FRE("") déclenche un TYPE MISMATCH ERROR.

```
D3D6  LDA 28       D47E  LDA 28       prendre type de l'argument
D3D8  BEQ D3DD     D480  BEQ D485     si numérique, on saute
D3DA  JSR $D715    D482  JSR $D7D0     si chaine, on enlève la réservation
D3DD  JSR $D595    D485  JSR $D650     en tous cas on réorganise les chaines
D3E0  SEC          D488  SEC          et on calcule l'espace disponible
D3E1  LDA A2       D489  LDA A2
D3E3  SBC A0       D48B  SBC A0       c'est à dire plus basse chaine
D3E5  TAY          D48D  TAY          moins plus haut tableau
D3E6  LDA A3       D48E  LDA A3
D3E8  SBC A1       D490  SBC A1       on a le résultat dans YA
.....              D492  LDX #00      on indique que le résultat
.....              D494  STX 28       est numérique
D3EA  JMP $D8D5    D496  JMP $DF40     YA --) ACC1 (non signé)
```

YA --) ACC1 (SIGNE)

```
D3ED  LDX #00      D499  LDX #00
D3EF  STX 28       D49B  STX 28       forcer résultat numérique
D3F1  STA D1       D49D  STA D1       sauver poids fort
D3F3  STY D2       D49F  STY D2       et poids faible
D3F5  LDX #90      D4A1  LDX #90      indiquer exposant
D3F7  JMP $DF1D    D4A3  JMP $DF2C    et ajuster la mantisse
```

'POS' (FONCTION)

Bogue: sur la V1.1, un cas donne un résultat érronné: si on veut le résultat du clavier et qu'on est sur l'imprimante, il aurait fallu faire pour le clavier la même chose que pour l'imprimante.

Compte tenu du fait que sur la V1.Ø, la fonction n'est pas prévue pour l'imprimante, il n'y a pas de bogue. Les programmeurs s'étaient sans doute déjà aperçus que #3Ø était décalé, puisque la valeur est prise en #269. De là à corriger la bogue...

```
.............. D4A6  JSR $D8CB   ACC1 --> X
.............. D4A9  TXA         et X dans A !
.............. D4AA  BEQ D4B4    si argument=Ø, c'est le clavier
.............. D4AC  LDY Ø258    sinon, c'est l'imprimante
.............. D4AF  BIT Ø2F1    mais au fait, sommes-nous sur l'imprimante ?

.............. D4B2  BPL D4B6    oui, c'est la bonne valeur
D3FA LDY Ø269 D4B4  LDY 3Ø      prendre valeur courante
```

Y --> ACC1 (non signé)

```
D3FD LDA #ØØ     D4B6 LDA #ØØ     poids fort=Ø
D3FF BEQ D3ED    D4B8 BEQ D499    et mettre résultat dans ACC1
```

'DEF' (COMMANDE)

```
D4Ø1 CMP #&USR   D4BA CMP #&USR   est-ce DEF USR ?
D4Ø3 BNE D426    D4BC BNE D4DF    non, on saute

Traiter DEF USR

D4Ø5 JSR $ØØE2   D4BE JSR $ØØE2   on saute USR
D4Ø8 LDA #&=     D4C1 LDA #&=     on demande un '='
D4ØA JSR $CFDB   D4C3 JSR $DØ67
D4ØD JSR $E79D   D4C6 JSR $E853   et on évalue l'expression
D41Ø LDA 33      D4C9 LDA 33      on récupère la valeur
D412 LDY 34      D4CB LDY 34      (inutile, elle est dans YA)
D414 STA 22      D4CD STA 22      et on modifie
D416 STY 23      D4CF STY 23      le vecteur
D418 RTS         D4D1 RTS
```

```
D419  LDX A9      D4D2  LDX A9      prendre numéro de ligne
D41B  INX         D4D4  INX         tester si #FF
D41C  BNE D3A4    D4D5  BNE D4D1    non,on sort
D41E  LDX #95     D4D7  LDX #95     oui,'ILLEGAL DIRECT'
D420  BYT #2C     D4D9  BYT #2C     sauter l'instruction suivante
D421  LDX #E5     D4DA  LDX #E5     'UNDEF'D FONCTION'
D423  JMP $C485   D4DC  JMP $C47E
```

Traiter DEF FN()

Principe: on saisit l'adresse de la variable, puis on empile les 5 caractères (adresse de la variable et adresse de la définition, ainsi qu'une valeur bidon. Cette valeur bidon est d'ailleurs le premier caractère de la définition). Ces valeurs sont ensuite dépilées et envoyées dans la définition de la fonction.

```
D426  JSR $D454   D4DF  JSR $D50D  prendre adresse de la fonction
D429  JSR $D419   D4E2  JSR $D4D2  interdire le mode direct
D42C  JSR $CFD6   D4E5  JSR $D062  demander '('
D42F  LDA #80     D4E8  LDA #80    indiquer pas d'entier
D431  STA 2B      D4EA  STA 2B
D433  JSR $D0FC   D4EC  JSR $D188  prendre adresse de la variable
D436  JSR $CE7A   D4EF  JSR $CF06  vérifier pas de chaine
D439  JSR $CFD3   D4F2  JSR $D05F  demander ')'
D43C  LDA #&=     D4F5  LDA #&=
D43E  JSR $CFDB   D4F7  JSR $D067  et demander '='
D441  PHA         D4FA  PHA        sauver 1er caractère de la définition
D442  LDA B7      D4FB  LDA B7     prendre adresse de la variable
D444  PHA         D4FD  PHA        et la sauver
D445  LDA B6      D4FE  LDA B6
D447  PHA         D500  PHA        idem pour le poids faible
D448  LDA EA      D501  LDA EA
D44A  PHA         D503  PHA        sauver TXTPTR
D44B  LDA E9      D504  LDA E9
D44D  PHA         D506  PHA        (c'est l'adresse de la définition.)
D44E  JSR $CA0A   D507  JSR $CA3C  aller à la fin de l'instruction,via DATA
D451  JMP $D4C2   D50A  JMP $D57D  et affecterla fonction.
```

Prendre l'adresse de la fonction

Entrée: TXTPTR pointe sur FN

Sortie: $BD-$BE contient l'adresse de la fonction.

```
D454  LDA #&FN    D50D  LDA #&FN    demander FN
D456  JSR $CFDB   D50F  JSR $D067
D459  ORA #80     D512  ORA #80     ajuster le code de la première lettre
..............    D514  LDX #80     interdire les entiers
D45B  STA 2B      D516  STX 2B
D45D  JSR $D183   D518  JSR $D18F   prendre adresse de la fonction
D460  STA BD      D51B  STA BD
D462  STY BE      D51D  STY BE      et la sauver
D464  JMP $CE7A   D51F  JMP $CF06   vérifier pas chaine
```

Evaluer une fonction

Principe: on saisit l'adresse de la fonction, que l'on sauve sur la pile pour évaluer le paramètre (pour le cas où ce paramètre contiendrait lui aussi des appels à une fonction). Ensuite, on sauve sur la pile la valeur de la variable, qui ne doit pas être affectée.

La variable prend ensuite la valeur du paramètre, et on évalue la fonction, que l'on sauve. Il ne reste plus qu'à récupérer ni vu ni connu l'ancienne valeur de la variable, qui apparait inchangée.

```
D467  JSR $D454   D522  JSR $D50D   trouver l'adresse de la fonction
D46A  LDA BE      D525  LDA BE      et sauver son adresse
D46C  PHA         D527  PHA
D46D  LDA BD      D528  LDA BD      idem poids faible
D46F  PHA         D52A  PHA
D470  JSR $CFCD   D52B  JSR $D059   évaluer la variable
D473  JSR $CE7A   D52E  JSR $CF06   vérifier numérique
D476  PLA         D531  PLA         récupérer adresse fonction
D477  STA BD      D532  STA BD
D479  PLA         D534  PLA
D47A  STA BE      D535  STA BE      et aussi poids fort
D47C  LDY #02     D537  LDY #02
D47E  LDA (BD),Y  D539  LDA (BD),Y  prendre adresse de la variable
D480  STA B6      D53B  STA B6      et la sauver
D482  TAX         D53D  TAX         dans X aussi
D483  INY         D53E  INY
D484  LDA (BD),Y  D53F  LDA (BD),Y  idem pour le poids fort
D486  BEQ D421    D541  BEQ D4DA     si 0,UNDEF'D FONCTION
D488  STA B7      D543  STA B7      sinon sauver aussi
D48A  INY         D545  INY         Y=4
D48B  LDA (B6),Y  D546  LDA (B6),Y  sauver la valeur de la variable sur la pile
```

```
D48D  PHA          D548  PHA          de la fonctions
D48E  DEY          D549  DEY
D48F  BPL D48B     D54A  BPL D546
D491  LDY B7       D54C  LDY B7       XY=adresse de la variable
D493  JSR $DEA5    D54E  JSR $DEAD    AACC1 --) (XY) (Y=Ø en sortant)
D496  LDA EA       D551  LDA EA       sauver TXTPTR
D498  PHA          D553  PHA
D499  LDA E9       D554  LDA E9
D49B  PHA          D556  PHA          et aussi poids faible
D49C  LDA (BD),Y   D557  LDA (BD),Y   prendre adresse de la définition
D49E  STA E9       D559  STA E9       dans TXTPTR
D4AØ  INY          D55B  INY
D4A1  LDA (BD),Y   D55C  LDA (BD),Y   poids fort aussi
D4A3  STA EA       D55E  STA EA
D4A5  LDA B7       D56Ø  LDA B7       sauver adresse de la fonction
D4A7  PHA          D562  PHA
D4A8  LDA B6       D563  LDA B6
D4AA  PHA          D565  PHA          poids faible aussi
D4AB  JSR $CE77    D566  JSR $CFØ3    évaluer la définition
D4AE  PLA          D569  PLA
D4AF  STA BD       D56A  STA BD       récupérer adresse de la fonction
D4B1  PLA          D56C  PLA
D4B2  STA BE       D56D  STA BE       en #BD-#BE cette fois
D4B4  JSR $ØØE8    D56F  JSR $ØØE8    la définition se termine correctement ?
D4B7  BEQ D4BC     D572  BEQ D577
D4B9  JMP $CFE4    D574  JMP $D97Ø    non,'SYNTAX ERROR'
D4BC  PLA          D577  PLA          oui,on récupère
D4BD  STA E9       D578  STA E9       TXTPTR pour reprendre le cours normal
D4BF  PLA          D57A  PLA          du programme
D4CØ  STA EA       D57B  STA EA       poids faible aussi
```

Affecter une fonction, récupérer la valeur initiale du paramètre

```
D4C2  LDY #ØØ      D57D  LDY #ØØ      indexer l'adresse de définition
D4C4  PLA          D57F  PLA          récupérer poids faible définition
D4C5  STA (BD),Y   D58Ø  STA (BD),Y   (ou l'exposant du paramètre) et sauver
D4C7  PLA          D582  PLA
D4C8  INY          D583  INY
D4C9  STA (BD),Y   D584  STA (BD),Y   idem pour le poids fort (ou Octet 1)
D4CB  PLA          D586  PLA
D4CC  INY          D587  INY          récupére adresse fonction (ou octet 2)
D4CD  STA (BD),Y   D588  STA (BD),Y   et la sauver
D4CF  PLA          D58A  PLA
D4DØ  INY          D58B  INY
```

```
D4D1  STA (BD),Y  D58C  STA (BD),Y  idem poids fort adresse (ou octet 3)
D4D3  PLA         D58E  PLA         récupérer bidon ou octet 4
D4D4  INY         D58F  INY
D4D5  STA (BD),Y  D590  STA (BD),Y  et le sauver,et c'est tout.
D4D7  RTS         D592  RTS
```

D) LA GESTION DES CHAINES

1-Evaluation et réservation

'STR$' (FONCTION)

Bogue: sur V1.0,le premier caractère est un #02 dans le cas d'un nombre positif, à cause de #E0D3.

Remarque: le nombre est stocké à partir de #00FF au lieu de #100, car une chaine qui débute en page 0 est considérée comme temporaire, ce qui doit être le cas pour un nombre.

```
D4D8  JSR $CE7A   D593  JSR $CF06  vérifier argument numérique
D4DB  LDY #00     D596  LDY #00    placer le nombre en #00FF
D4DD  JSR $E0D3   D598  JSR $E0D7  et faire la conversion
D4E0  PLA         D59B  PLA        dépiler adresse de retour
D4E1  PLA         D59C  PLA        pour éviter vérification numérique
D4E2  LDA #FF     D59D  LDA #FF    (Cf #D042/#D0CE)
D4E4  LDY #00     D59F  LDY #00    AY=adresse de la chaine
D4E6  BE0 D4FA    D5A1  BE0 D5B5   inconditionnel:et évaluer la chaine

D4E8  LDX D3      D5A3  LDX D3
D4EA  LDY D4      D5A5  LDY D4     prendre adresse du descripteur
D4EC  STX BF      D5A7  STX BF
D4EE  STY C0      D5A9  STY C0     et la sauver en #BF-#C0
D4F0  JSR $D563   D5AB  JSR $D61E  réserver une chaine de longueur A
D4F3  STX D1      D5AE  STX D1     sauver adresse de la chaine
D4F5  STY D2      D5B0  STY D2
D4F7  STA D0      D5B2  STA D0     et aussi sa longueur
D4F9  RTS         D5B4  RTS
```

EVALUER UNE CHAINE POINTEE PAR AY

Remarque: c'est la routine centrale de la gestion des chaines, puisque c'est elle qui empile les descripteurs dans la pile des descripteurs.

Principe: la longueur de la chaine est évaluée. ensuite, si la chaine est dans la page Ø, c'est à dire mode direct ou nombre évalué par STR$, une zone est réservée, et la chaine y est transférée.

Dans tous les cas, le descripteur est empilé dans la pile des descripteurs.

```
D4FA  LDX #'"'     D5B5  LDX #'"'     prendre code du guillemet
D4FC  STX 24       D5B7  STX 24
D4FE  STX 25       D5B9  STX 25
D500  STA DE       D5BB  STA DE
D502  STY DF       D5BD  STY DF       sauver adresse de début pour pointeur
D504  STA D1       D5BF  STA D1
D506  STY D2       D5C1  STY D2       et aussi pour référence dans ACC1
D508  LDY #FF      D5C3  LDY #FF      initialiser la longueur
D50A  INY          D5C5  INY          caractère suivant
D50B  LDA (DE),Y   D5C6  LDA (DE),Y   prendre caractère dans la chaine
D50D  BEQ D51B     D5C8  BEQ D5D6     si Ø,c'est fini (premier terminateur)
D50F  CMP 24       D5CA  CMP 24
D511  BEQ D517     D5CC  BEQ D5D2
D513  CMP 25       D5CE  CMP 25
D515  BNE D50A     D5D0  BNE D5C5     si aucun des deux,on continue
D517  CMP #'"'     D5D2  CMP #'"'     est-on sorti à cause d'un '"' ?
D519  BEQ D51C     D5D4  BEQ D5D7     oui,sauter,C=1
D51B  CLC          D5D6  CLC          non,C=Ø
D51C  STY D0       D5D7  STY D0       sauver longueur
D51E  TYA          D5D9  TYA
D51F  ADC DE       D5DA  ADC DE       et ajouter au début (+1 si guillemet)
D521  STA E0       D5DC  STA E0       et on sauve l'adresse de la fin
D523  LDX DF       D5DE  LDX DF
D525  BCC D528     D5E0  BCC D5E3
D527  INX          D5E2  INX          on ajuste aussi le poids fort
D528  STX E1       D5E3  STX E1       sauver aussi...
D52A  LDA DF       D5E5  LDA DF       poids fort du début
D52C  BNE D539     D5E7  BNE D5F4     sauter si programme ou ROM
D52E  TYA          D5E9  TYA          si direct ou décimal A=longueur
D52F  JSR $D4E8    D5EA  JSR $D5A3    on réserve une chaine temporaire
D532  LDX DE       D5ED  LDX DE       XY=adresse du début
D534  LDY DF       D5EF  LDY DF
D536  JSR $D6F7    D5F1  JSR $D7B2    on transfère la chaine --> la zone réservée
```

```
D539  LDX 85      D5F4  LDX 85      on empile: prendre le pointeur
D53B  CPX #91     D5F6  CPX #91     la pile est-elle pleine ?
D53D  BNE D544    D5F8  BNE D5FF    non,on saute
D53F  LDX #C4     D5FA  LDX #C4     oui,'FORMULA TOO COMPLEX ERROR'
D541  JMP $C485   D5FC  JMP $C47E
D544  LDA D0      D5FF  LDA D0      prendre longueur
D546  STA 00,X    D601  STA 00,X    et sauver dans la pile
D548  LDA D1      D603  LDA D1      prendre poids faible adresse
D54A  STA 01,X    D605  STA 01,X    et sauver aussi
D54C  LDA D2      D607  LDA D2      de même pour le poids fort
D54E  STA 02,X    D609  STA 02,X
D550  LDY #00     D60B  LDY #00     adresse du descripteur dans XY
D552  STX D3      D60D  STX D3
D554  STY D4      D60F  STY D4      et on sauve nouvelle adresse du descripteur
D556  STY DF      D611  STY DF
D558  DEY         D613  DEY         Y=#FF
D559  STY 28      D614  STY 28      indiquer type chaine
D55B  STX 86      D616  STX 86      sauver ancien pointeur de pile
D55D  INX         D618  INX
D55E  INX         D619  INX
D55F  INX         D61A  INX         calculer le nouveau
D560  STX 85      D61B  STX 85      et le sauver
D562  RTS         D61D  RTS
```

RESERVER UNE CHAINE DE LONGUEUR A

Entrée: A contient la longueur de la chaine à réserver.

Sortie: s'il y a la place, #A4-#A5 pointe sur le début de la zone réservée, de même que XY et #A2-#A3.
A est conservé.

Principe: limpide. On essaye de placer la chaine en dessous du plafond. S'il y a de la place, c'est gagné, il faut sinon tenter une réorganisation, après quoi c'est désespéré...

```
D563  LSR 2A      D61E  LSR 2A      indiquer pas encore de réorganisation
D565  PHA         D620  PHA         sauver la longueur
D566  EOR #FF     D621  EOR #FF     et complémenter
D568  SEC         D623  SEC         vrai complément (ajouter 1)
D569  ADC A2      D624  ADC A2      soustraire longueur du plafond
D56B  LDY A3      D626  LDY A3      y compris le poids fort
D56D  BCS D570    D628  BCS D62B
```

```
D56F  DEY          D62A  DEY          dans Y
D57Ø  CPY A1       D62B  CPY A1       au dessus des tableaux ?
D572  BCC D585     D62D  BCC D64Ø     non, traiter l'erreur
D574  BNE D57A     D62F  BNE D635     oui, c'est bon
D576  CMP AØ       D631  CMP AØ       on sait pas: voyons le poids faible
D578  BCC D585     D633  BCC D64Ø     c'est sur,pas de place
D57A  STA A2       D635  STA A2       placer nouvelle adresse
D57C  STY A3       D637  STY A3       de la plus basse chaine
D57E  STA A4       D639  STA A4
D58Ø  STY A5       D63B  STY A5       et aussi adresse de la chaine crée
D582  TAX          D63D  TAX          adresse aussi dans XY
D583  PLA          D63E  PLA          on reprend la longueur
D584  RTS          D63F  RTS

D585  LDX #4D      D64Ø  LDX #4D      indexer 'OUT OF MEMORY'
D587  LDA 2A       D642  LDA 2A       réorganisation faite ?
D589  BMI D541     D644  BMI D5FC     oui, plus d'espoir: erreur
D58B  JSR $D595    D646  JSR $D65Ø    non, on tente une réorganisation
D58E  LDA #8Ø      D649  LDA #8Ø      on indique qu'elle a été faite
D59Ø  STA 2A       D64B  STA 2A       (SEC:ROR aurait été mieux)
D592  PLA          D64D  PLA          on récupère la longueur
D593  BNE D565     D64E  BNE D62Ø     inconditionnel: réessayer
```

2-Réorganisation des chaines

REORGANISER LES CHAINES

Entrée: #C2=Ø

Sortie: Chaine réorganisées, et #A2 initialisé donc.

Remarque: La souplesse d'emploi des variables alphanumériques sous BASIC est liée entièrement à la gestion dynamique des chaines. Cette gestion dynamique, c'est cette routine.
Le problème: A chaque affectation de variable alpha, une nouvelle zone est réservée, en dessous du plafond, le plafond étant descendu.
Petit à petit, la zone des chaines se rempli de chaines inutiles, qui sont des valeurs temporaires ou d'anciennes valeurs. Il vient un moment où, à force de descendre, les chaines voudraient aller plus bas que la zone des tableaux. Il faut alors réorganiser les chaines, c'est à dire éliminer les chaines inutiles, ou plutot ne garder que celles utiles.

Principe: les seules chaines utiles sont celles qui ont un descripteur: soit dans la zone des variables, dans la zone des tableaux, ou même dans la pile des descripteurs.

Le principe général est simple: on va se fixer un plafond temporaire (le HIMEM au début), et chercher la chaine qui s'en rapproche le plus, sans être au dessus toutefois, car elle aura été traitée lors de la passe précédente.

Lorsqu'on a trouvé la chaine correspondante, on la déplace de manière à la coller au plafond temporaire, et on place le nouveau plafond temporaire juste en dessous, et on recommence.

Pour être sur que la chaine est bien la plus haute, il faut décrire toutes les chaines à chaque étape. Le temps d'exécution sera donc globalement proportionnel au carré du nombre de chaines, ce qui explique la lenteur de la réorganisation.

Si on n'a trouvé aucune chaine correspondant aux critères, on arrête la réorganisation.

Le principe général peut s'illustrer par le schéma suivant (Figure D4-A).

Voici la signification des différentes variables employées:

$A2-$A4: plafond temporaire, évoluant peu à peu vers le plafond définitif.
$CE-$CF: adresse de la plus haute chaine (en dessous du plafond) courante. Ce pointeur est initialisé à la fin des tableaux et non à Ø, pour éviter aux chaines stockées dans le texte Basic lui même.
$BD-$BE: adresse du descripteur de la plus haute variable courante. C'est lui qui sert aussi de drapeau pour savoir si la réorganisation est finie: onitialisé à Ø, il n'est touché que lorsqu'un chaine correspondants aux critères est trouvée.
$91-$92: pointeur des descripteurs.
$C2: longueur d'un élément: 3 pour la pile des descripteurs, 7 pour les variables,3 pour les tableaux. Considéré au début comme valant Ø. (voir initialisation basic)
$C4: longueur de l'élément de la plus haute chaine courante.

Voici de plus un organigramme de la routine

Figure D4-B

(voir page 170)

```
D595 LDX A6      D650 LDX A6
D597 LDA A7      D652 LDA A7      prendre HIMEM
D599 STX A2      D654 STX A2
D59B STA A3      D656 STA A3      comme plafond courant
```

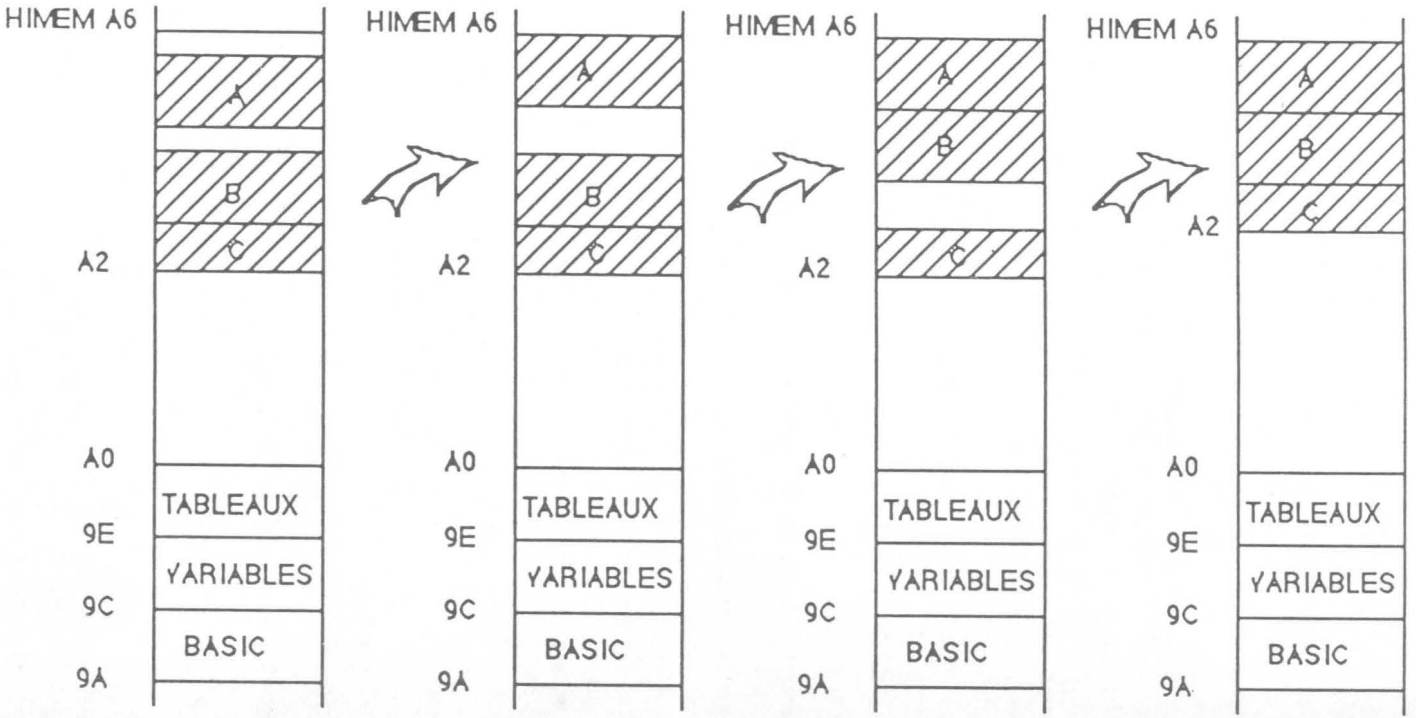

HIMEM A6
A
B
C
A2
A0
TABLEAUX
9E
VARIABLES
9C
BASIC
9A
HIMEM A6
A
B
C
A2
A0
TABLEAUX
9E
VARIABLES
9C
BASIC
9A
HIMEM A6
A
B
C
A2
A0
TABLEAUX
9E
VARIABLES
9C
BASIC
9A
HIMEM A6
A
B
C
A2
A0
TABLEAUX
9E
VARIABLES
9C
BASIC
9A

Figure D4-A

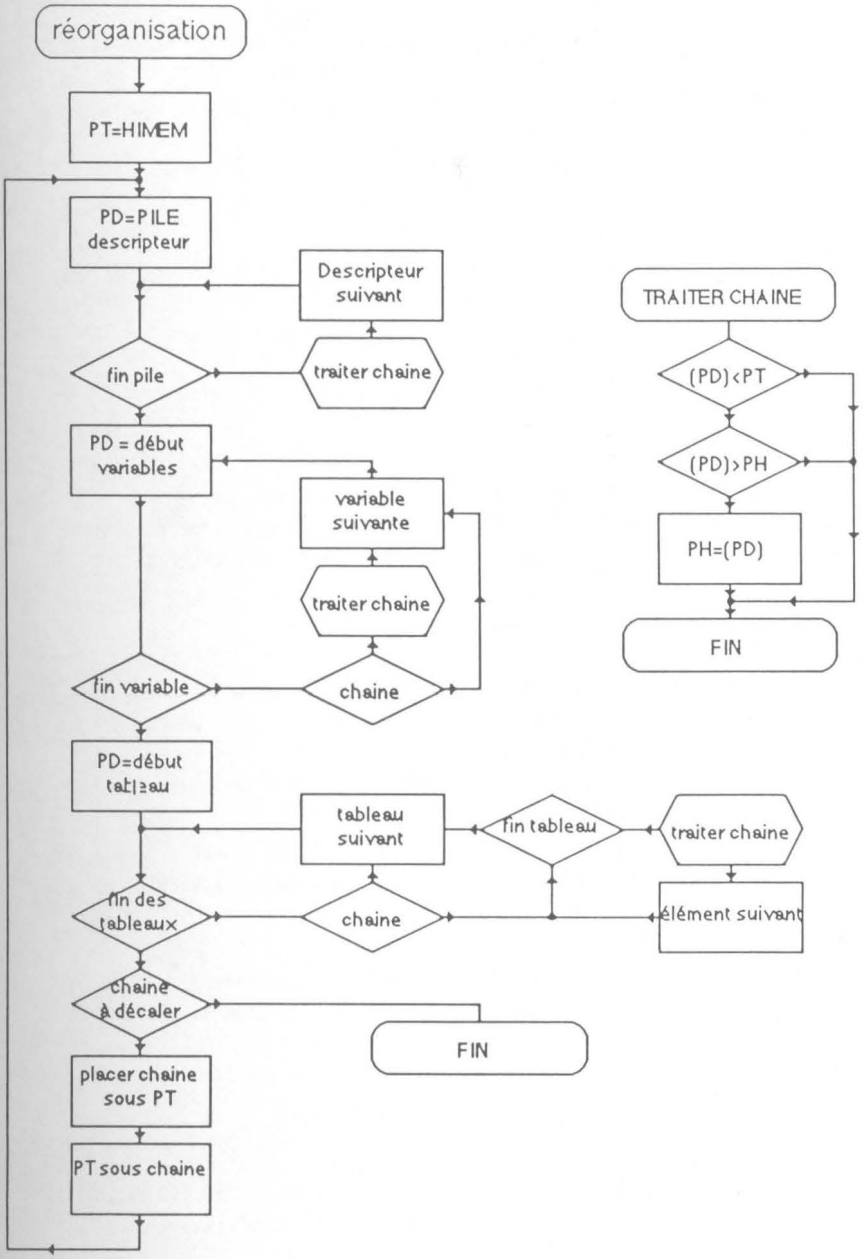

réorganisation
PT=HIMEM
PD=PILE descripteur
Descripteur suivant
fin pile
traiter chaine
PD = début variables
variable suivante
traiter chaine
fin variable
chaine
PD=début tableau
tableau suivant
fin tableau
traiter chaine
fin des tableaux
chaine
élément suivant
chaine à décaler
FIN
placer chaine sous PT
PT sous chaine
TRAITER CHAINE
(PD)<PT
(PD)>PH
PH=(PD)
FIN

Figure D4-B

```
D59D  LDY #ØØ     D658  LDY #ØØ
D59F  STY BE      D65A  STY BE      adresse de la plus haute variable
D5A1  STY BD      D65C  STY BD      à Ø pour l'instant
D5A3  LDA AØ      D65E  LDA AØ      prendre fin des tableaux
D5A5  LDX A1      D66Ø  LDX A1
D5A7  STA CE      D662  STA CE      plus basse chaine courante
D5A9  STX CF      D664  STX CF
```

Décrire la pile des descripteurs

```
D5AB  LDA #88     D666  LDA #88     AX=pile des descripteurs
D5AD  LDX #ØØ     D668  LDX #ØØ
D5AF  STA 91      D66A  STA 91      dans pointeur de travail
D5B1  STX 92      D66C  STX 92
D5B3  CMP 85      D66E  CMP 85      fin de la pile atteinte ?
D5B5  BEQ D5BC    D67Ø  BEQ D677    oui,passer aux variables
D5B7  JSR $D636   D672  JSR $D6F1   non,chercher la plus près du plafond
D5BA  BEQ D5B3    D675  BEQ D66E     inconditionnel:finir descripteurs
```

Décrire les variables

```
D5BC  LDA #Ø7     D677  LDA #Ø7     initialiser la longueur
D5BE  STA C2      D679  STA C2      d'une variable
D5CØ  LDA 9C      D67B  LDA 9C
D5C2  LDX 9D      D67D  LDX 9D      prendre début des variables
D5C4  STA 91      D67F  STA 91
D5C6  STX 92      D681  STX 92      comme pointeur de travail
D5C8  CPX 9F      D683  CPX 9F      est-ce la fin des variables ?
D5CA  BNE D5DØ    D685  BNE D68B    non,on continue
D5CC  CMP 9E      D687  CMP 9E
D5CE  BEQ D5D5    D689  BEQ D69Ø    oui,décrire les tableaux
D5DØ  JSR $D62C   D68B  JSR $D6E7   chercher la plus près du plafond
D5D3  BEQ D5C8    D68E  BEQ D683    incontitionnel
```

Décrire les tableaux

```
D5D5  STA C7      D69Ø  STA C7      début des tableaux (fin des variables)
D5D7  STX C8      D692  STX C8      dans le pointeur de travail
D5D9  LDA #Ø3     D694  LDA #Ø3
D5DB  STA C2      D696  STA C2      initiliser longueur d'un élément
D5DD  LDA C7      D698  LDA C7
D5DF  LDX C8      D69A  LDX C8      prendre adresse du tableau courant
D5E1  CPX A1      D69C  CPX A1      est-ce la fin ?
```

```
D5E3  BNE D5EC     D69E  BNE D6A7     non,on continue
D5E5  CMP A0       D6A0  CMP A0
D5E7  BNE D5EC     D6A2  BNE D6A7
D5E9  JMP $D675    D6A4  JMP $D730     oui, aller finir

Tester un tableau

D5EC  STA 91       D6A7  STA 91       sauver début du tableau
D5EE  STX 92       D6A9  STX 92       dans pointeur de travail
D5F0  LDY #00      D6AB  LDY #00      préparer l'index
D5F2  LDA (91),Y   D6AD  LDA (91),Y   premier caractère du nom
D5F4  TAX          D6AF  TAX          dans X
D5F5  INY          D6B0  INY
D5F6  LDA (91),Y   D6B1  LDA (91),Y   le deuxième
D5F8  PHP          D6B3  PHP          sauver son signe suffit
D5F9  INY          D6B4  INY
D5FA  LDA (91),Y   D6B5  LDA (91),Y   longueur du tableau poids faible
D5FC  ADC C7       D6B7  ADC C7       et on calcule l'adresse du prochain tableau
D5FE  STA C7       D6B9  STA C7       on resauve
D600  INY          D6BB  INY
D601  LDA (91),Y   D6BC  LDA (91),Y   idem pour le poids fort
D603  ADC C8       D6BE  ADC C8
D605  STA C8       D6C0  STA C8
D607  PLP          D6C2  PLP
D608  BPL D5DD     D6C3  BPL D698     on passe au suivant si tableau réel
D60A  TXA          D6C5  TXA
D60B  BMI D5DD     D6C6  BMI D698     de même si tableau entier
D60D  INY          D6C8  INY          indexer la dimension
D60E  LDA (91),Y   D6C9  LDA (91),Y   prendre la dimension
D610  LDY #00      D6CB  LDY #00      et calculer la place de l'entête
D612  ASL A        D6CD  ASL A        x2 (2 octets pour chaque composante)
D613  ADC #05      D6CE  ADC #05      +5 (constante:2 nom,2 longueur,1 dimension
D615  ADC 91       D6D0  ADC 91       et on ajuste
D617  STA 91       D6D2  STA 91       le pointeur de travail
D619  BCC D61D     D6D4  BCC D6D8     au début effectif du tableau
D61B  INC 92       D6D6  INC 92       sans oublier le poids fort
D61D  LDX 92       D6D8  LDX 92       tester si on est à la fin du tableau
D61F  CPX C8       D6DA  CPX C8
D621  BNE D627     D6DC  BNE D6E2     non, on continue
D623  CMP C7       D6DE  CMP C7
D625  BEQ D5E1     D6E0  BEQ D69C     oui, passer au tableau suivant
D627  JSR $D636    D6E2  JSR $D6F1    non, traiter l'élément
D62A  BEQ D61F     D6E5  BEQ D6DA     et passer au suivant
```

```
Traiter une variable

D62C  LDA (91),Y  D6E7  LDA (91),Y  premier caractère du nom
D62E  BMI D665    D6E9  BMI D720    c'est un entier,passer à la suivante
D630  INY         D6EB  INY
D631  LDA (91),Y  D6EC  LDA (91),Y  deuxième
D633  BPL D665    D6EE  BPL D720    c'est un réel, variable suivante aussi.
D635  INY         D6F0  INY         ajuster Y sur le descripteur

Traiter l'élément pointé par #91-#92
,indexé par Y

D636  LDA (91),Y  D6F1  LDA (91),Y  prendre la longueur
D638  BEQ D665    D6F3  BEQ D720    chaine vide, on passe à la suite
D63A  INY         D6F5  INY
D63B  LDA (91),Y  D6F6  LDA (91),Y  adresse poids faible
D63D  TAX         D6F8  TAX         dans X
D63E  INY         D6F9  INY
D63F  LDA (91),Y  D6FA  LDA (91),Y  adresse poids fort
D641  CMP A3      D6FC  CMP A3      vérifier au dessous du plafond temporaire
D643  BCC D64B    D6FE  BCC D706    oui, on continue
D645  BNE D665    D700  BNE D720    non, suivante
D647  CPX A2      D702  CPX A2      pas si sur, on teste le poids fort
D649  BCS D665    D704  BCS D720    trop haute, suivante
D64B  CMP CF      D706  CMP CF      et par rapport à la plus haute ?
D64D  BCC D665    D708  BCC D720    en dessous, on sort
D64F  BNE D655    D70A  BNE D710    au dessus,elle remporte le titre
D651  CPX CE      D70C  CPX CE      on sait pas, voyons le poids faible
D653  BCC D665    D70E  BCC D720    trop bas: suivante
D655  STX CE      D710  STX CE      c'est la variable la plus haute
D657  STA CF      D712  STA CF      sous le plafond
D659  LDA 91      D714  LDA 91      on récupère l'adresse
D65B  LDX 92      D716  LDX 92      du descripteur
D65D  STA BD      D718  STA BD
D65F  STX BE      D71A  STX BE      et on le sauve
D661  LDA C2      D71C  LDA C2      on sauve aussi la longueur de l'élément
D663  STA C4      D71E  STA C4

Elément suivant

D665  LDA C2      D720  LDA C2      prendre longueur de l'élément
D667  CLC         D722  CLC
D668  ADC 91      D723  ADC 91      ajouté à l'adresse courante
D66A  STA 91      D725  STA 91
```

```
D66C  BCC D67Ø     D727  BCC D72B    poids fort aussi
D66E  INC 92       D729  INC 92      on obtient la nouvelle adresse
D67Ø  LDX 92       D72B  LDX 92      que l'on a aussi dans AX
D672  LDY #ØØ      D72D  LDY #ØØ      index à Ø pour la suite
D674  RTS          D72F  RTS
```

Continuer la réorganisation

```
D675  LDA BE       D73Ø  LDA BE      y-a-t-il une chaine à déplacer ?
D677  ORA BD       D732  ORA BD
D679  BEQ D67Ø     D734  BEQ D72B     non, c'est fini (ouf !)
D67B  LDA C4       D736  LDA C4      oui, on prend la longueur de l'élément
D67D  AND #Ø4      D738  AND #Ø4     variable: 4 tableau: Ø
D67F  LSR A        D73A  LSR A       variable: 2 tableau: Ø
D68Ø  TAY          D73B  TAY         on calcule l'index du descripteur
D681  STA C4       D73C  STA C4      et on le sauve
D683  LDA (BD),Y   D73E  LDA (BD),Y  prendre la longueur
D685  ADC CE       D74Ø  ADC CE      on l'ajoute à l'adresse de la chaine
D687  STA C9       D742  STA C9      c'est le dernier octet
D689  LDA CF       D744  LDA CF      du bloc á décaler
D68B  ADC #ØØ      D746  ADC #ØØ
D68D  STA CA       D748  STA CA
D68F  LDA A2       D74A  LDA A2      plafond temporaire
D691  LDX A3       D74C  LDX A3
D693  STA C7       D74E  STA C7      comme adresse cible
D695  STX C8       D75Ø  STX C8      (#CE=bas de la chaine)
D697  JSR $C3FF    D752  JSR $C3FB   décaler la chaine vers le plafond
D69A  LDY C4       D755  LDY C4      prendre index du descripteur
D69C  INY          D757  INY         on ajuste sur l'adresse de la chaine
D69D  LDA C7       D758  LDA C7      prendre nouvelle adresse poids faible
D69F  STA (BD),Y   D75A  STA (BD),Y  et la stocker
D6A1  TAX          D75C  TAX         sauver dans X
D6A2  INC C8       D75D  INC C8      ajuster poids fort
D6A4  LDA C8       D75F  LDA C8      le prendre
D6A6  INY          D761  INY
D6A7  STA (BD),Y   D762  STA (BD),Y  et le sauver
D6A9  JMP $D599    D764  JMP $D654   et recommencer...
```

'+' (OPERATEUR) :CONCATENATION DE CHAINES

Principe: après avoir repris les divers descripteurs, on calcule la nouvelle longueur de la chaine, et on réserve la place pour cette chaine. L'adresse du début de cette zone est en #A4-#A5 comme d'habitude.

On y transfère alors la première chaine. #A4-#A5 pointe alors sur la fin de la chaine, il suffit de transférer la 2ème chaine.

```
D6AC  LDA D4      D767  LDA D4      Sauver sur la pile
D6AE  PHA         D769  PHA         l'adresse du descripteur
D6AF  LDA D3      D76A  LDA D3      de l'opérande de gauche.
D6B1  PHA         D76C  PHA
D6B2  JSR $CF74   D76D  JSR $D000   évaluer opérande de droite
D6B5  JSR $CE7C   D770  JSR $CF08   et vérifier que c'est une chaine
D6B8  PLA         D773  PLA         récuperer adresse descripteur
D6B9  STA DE      D774  STA DE      de l'opérande de gauche
D6BB  PLA         D776  PLA         et la sauver en #DE-#DF pour indirection
D6BC  STA DF      D777  STA DF
D6BE  LDY #00     D779  LDY #00
D6C0  LDA (DE),Y  D77B  LDA (DE),Y  prendre longueur de la 1ère chaine
D6C2  CLC         D77D  CLC
D6C3  ADC (D3),Y  D77E  ADC (D3),Y  et l'ajouter à la longueur de la 2ème
D6C5  BCC D6CC    D780  BCC D787    si supérieur à 255,
D6C7  LDX #B5     D782  LDX #B5     'STRING TOO LONG ERROR'
D6C9  JMP $C485   D784  JMP $C47E   et l'afficher.
D6CC  JSR $D4E8   D787  JSR $D5A3   trouver A octets libres
D6CF  JSR $D6E9   D78A  JSR $D7A4   et y transferer l'opérande de gauche
D6D2  LDA BF      D78D  LDA BF      récupérer adresse descripteur
D6D4  LDY C0      D78F  LDY C0      de l'opérande de droite,
D6D6  JSR $D719   D791  JSR $D7D4   et enlever la réservation le concernant
D6D9  JSR $D6FB   D794  JSR $D7B6   transférer maintenant le 2ème opérande
D6DC  LDA DE      D797  LDA DE
D6DE  LDY DF      D799  LDY DF      prendre adresse du descripteur 1er opérande
D6E0  JSR $D719   D79B  JSR $D7D4   et enlever la réservation le concernant
D6E3  JSR $D539   D79E  JSR $D5F4   sauver descipteur sur la pile
D6E6  JMP $CEA5   D7A1  JMP $CF31   et continuer l'évaluation
```

TRANSFERT D'UNE CHAINE DE DESCRIPTEUR EN #DE-#DF VERS #A4-#A5

Entrée: une chaine a été réservée et l'adresse de l'emplacement libre est donc pointé par #A4-#A5.

#DE-#DF contient l'adresse du descripteur de la chaine à y déplacer.

Sortie: #A4-#A5 pointe juste après la chaine, #91-#92 pointe sur le début de la chaine origine.

```
D6E9  LDY #00     D7A4  LDY #00
D6EB  LDA (DE),Y  D7A6  LDA (DE),Y  Prendre longueur
```

```
D6ED  PHA          D7A8  PHA          et la sauver
D6EE  INY          D7A9  INY
D6EF  LDA (DE),Y   D7AA  LDA (DE),Y
D6F1  TAX          D7AC  TAX          prendre poids faible adresse
D6F2  INY          D7AD  INY          dans X
D6F3  LDA (DE),Y   D7AE  LDA (DE),Y   et poids fort dans A
D6F5  TAY          D7B0  TAY          puis Y
D6F6  PLA          D7B1  PLA          et récuperer longueur dans A
```

TRANSFERT D'UNE CHAINE POINTEE PAR XY ET DE LONGUEUR A

Entrée: XY contient l'adresse de la chaine, et A sa longueur. Pour le reste, voir la routine précédente.

Sortie: voir routine précédente.

```
D6F7  STX 91       D7B2  STX 91       sauver adresse de la chaine à transferer
D6F9  STY 92       D7B4  STY 92       dans #91-#92
D6FB  TAY          D7B6  TAY          et la longueur dans Y
D6FC  BEQ D708     D7B7  BEQ D7C3     si chaine vide, sortir
D6FE  PHA          D7B9  PHA          sauver la longueur pour addition future
D6FF  DEY          D7BA  DEY          caractère précédent (1ère fois: ajuster)
D700  LDA (91),Y   D7BB  LDA (91),Y   prendre caractère
D702  STA (A4),Y   D7BD  STA (A4),Y   et le transférer
D704  TYA          D7BF  TYA          etait-ce le dernier ?
D705  BNE D6FF     D7C0  BNE D7BA     non,continuer
D707  PLA          D7C2  PLA          récupérer la longueur
D708  CLC          D7C3  CLC          et l'ajouter pour que
D709  ADC A4       D7C4  ADC A4       #A4-#A5 pointe après la fin de la chaine
D70B  STA A4       D7C6  STA A4
D70D  BCC D711     D7C8  BCC D7CC
D70F  INC A5       D7CA  INC A5
D711  RTS          D7CC  RTS
```

ENLEVER LA RESERVATION,ET VERIFIER CHAINE

```
D712  JSR $CE7C    D7CD  JSR $CF08    Vérifier chaine
```

ENLEVER LA RESERVATION,ADRESSE DESCRIPTEUR DANS #D3-#D4

```
D715  LDA D3       D7D0  LDA D3       prendre adresse descripteur
```

```
D717  LDY D4      D7D2  LDY D4      dans AY
```

ENLEVER LA RESERVATION,ADRESSE DESCRIPTEUR DANS AY

Entrée: AY contient l'adresse du descripteur de la chaine

Sortie: XY et #91-#92 contiennent l'adresse de la chaine, A sa longueur. Si le descripteur était sur la pile des descripteurs, il en aura été enlevé.

Principe: Pour rendre la chaine utilisable, il faut donner son adresse et sa longueur. De plus, si le descripteur était dans la pile (chaine temporaire), on va l'enlever. Et de plus si, dans ce cas, la chaine était la plus basse, on remonte le plafond des chaines, pour ne pas encombrer la mémoire avec des chaines inutiles.

```
D719  STA 91      D7D4  STA 91      et la sauver
D71B  STY 92      D7D6  STY 92      pour indirection
D71D  JSR $D74A   D7D8  JSR $D8Ø5   ajuster pile des descripteurs
D72Ø  PHP         D7DB  PHP         sauver Z (=1 si la pile est descendue)
D721  LDY #ØØ     D7DC  LDY #ØØ
D723  LDA (91),Y  D7DE  LDA (91),Y  prendre longueur de la chaine
D725  PHA         D7EØ  PHA         et la sauver sur la pile
D726  INY         D7E1  INY
D727  LDA (91),Y  D7E2  LDA (91),Y  prendre adresse poids faible
D729  TAX         D7E4  TAX         dans X
D72A  INY         D7E5  INY
D72B  LDA (91),Y  D7E6  LDA (91),Y  puis poids fort
D72D  TAY         D7E8  TAY         dans Y
D72E  PLA         D7E9  PLA         récupérer longueur
D72F  PLP         D7EA  PLP         récupérer l'indicateur de chaine temporaire
D73Ø  BNE D745    D7EB  BNE D8ØØ    et sauter si ce n'en était pas une
D732  CPY A3      D7ED  CPY A3      était-ce la chaine la plus basse ?
D734  BNE D745    D7EF  BNE D8ØØ    non,c'est bon
D736  CPX A2      D7F1  CPX A2
D738  BNE D745    D7F3  BNE D8ØØ    tester poids faible aussi
D73A  PHA         D7F5  PHA         sauver longueur
D73B  CLC         D7F6  CLC         et ajouter à bas des chaines
D73C  ADC A2      D7F7  ADC A2
D73E  STA A2      D7F9  STA A2      pour calculer le nouveau plafond
D74Ø  BCC D744    D7FB  BCC D7FF    des chaines,sans oublier le poids fort
D742  INC A3      D7FD  INC A3
D744  PLA         D7FF  PLA         récupérer la longueur
D745  STX 91      D8ØØ  STX 91      sauver adresse de la chaine
```

```
D747  STY 92      D8Ø2  STY 92
D749  RTS         D8Ø4  RTS
```

DECREMENTER SI NECESSAIRE LA PILE DES DESCRIPTEURS

Entrée: AY=adresse du descripteur

Sortie: Z=1 si on décrémenté la pile, c'est à dire enlever effectivement une réservation temporaire.

```
D74A  CPY 87      D8Ø5  CPY 87      le descripteur est-il dans la pile ?
D74C  BNE D75A    D8Ø7  BNE D815    non, on sort, Z=Ø
D74E  CMP 86      D8Ø9  CMP 86      tester poids faible aussi
D75Ø  BNE D75A    D8ØB  BNE D815    non, on sort, Z=Ø
D752  STA 85      D8ØD  STA 85      sauver nouveau pointeur
D754  SBC #Ø3     D8ØF  SBC #Ø3     ajuster descripteur courant
D756  STA 86      D811  STA 86      et sauver
D758  LDY #ØØ     D813  LDY #ØØ     Z=1
D75A  RTS         D815  RTS
```

'CHR$' (Fonction)

```
D75B  JSR $D81Ø   D816  JSR $D8CB   ACC1 --) X
D75E  TXA         D819  TXA         et sauver le code ASCII
D75F  PHA         D81A  PHA
D76Ø  LDA #Ø1     D81B  LDA #Ø1     réserver une chaine de longueur 1
D762  JSR $D4FØ   D81D  JSR $D5AB
D765  PLA         D82Ø  PLA         récupérer le code
D766  LDY #ØØ     D821  LDY #ØØ
D768  STA (D1),Y  D823  STA (D1),Y  et le placer dans la chaine
D76A  PLA         D825  PLA         dépiler adresse de retour pour éviter
D76B  PLA         D826  PLA         le test de valeur numérique
D76C  JMP $D539   D827  JMP $D5F4   et sauver le descripteur sur la pile
```

'LEFT$' (Fonction)

Principe: trois fonctions se ramènent en fait à LEFT$: LEFT$,RIGHT$ et MID$. En fait, deux paramètres, que l'on obtient différemment selon le cas, définissent l'opératon à effectuer sur la chaine en argument:

Le premier c'est la position à laquelle commence la chaine résultat, et le deuxième sa longueur.

```
D76F  JSR $D7D0    D82A  JSR $D88B   récupérer les paramètres
D772  CMP (BF),Y   D82D  CMP (BF),Y  comparer longueur demandée avec possible
D774  TYA          D82F  TYA         A=0
D775  BCC D77B     D830  BCC D836    si plus petite, c'est bon
D777  LDA (BF),Y   D832  LDA (BF),Y  si plus grande,on prend
D779  TAX          D834  TAX         la longueur de la chaine, dans X aussi
D77A  TYA          D835  TYA         sauver index de début
D77B  PHA          D836  PHA
D77C  TXA          D837  TXA         et sauver longueur à transferer
D77D  PHA          D838  PHA
D77E  JSR $D4F0    D839  JSR $D5AB  réserver une chaine de longueur A
D781  LDA BF       D83C  LDA BF
D783  LDY C0       D83E  LDY C0      prendre adresse du descripteur de la chaine
D785  JSR $D719    D840  JSR $D7D4   et enlever réservation
D788  PLA          D843  PLA         récupérer longueur du transfert
D789  TAY          D844  TAY         dans Y
D78A  PLA          D845  PLA         et premier octet à déplacer
D78B  CLC          D846  CLC
D78C  ADC 91       D847  ADC 91      ajuster adresse de la chaine
D78E  STA 91       D849  STA 91      sur le premier octet à transferer
D790  BCC D794     D84B  BCC D84F
D792  INC 92       D84D  INC 92      poids fort aussi
D794  TYA          D84F  TYA         longueur dans A
D795  JSR $D6FB    D850  JSR $D7B6   et transfert vers zone réservée
D798  JMP $D539    D853  JMP $D5F4   sauver le descripteur sur la pile
```

'RIGHT$' (Fonction)

Principe: on calcule la position du début de la chaine à recopier. Le principe de la complémentation est utilisé pour donner à C une valeur permettant d'effectuer sur un éventuel dépassement de longueur le même test que pour LEFT$, et donc ne pas répéter ce test.

```
D79B  JSR $D7D0    D856  JSR $D88B   récupérer les paramètres
D79E  CLC          D859  CLC         calculer début de la chaine à tranférer
D79F  SBC (BF),Y   D85A  SBC (BF),Y  on enlève la longueur totale de la chaine
D7A1  EOR #FF      D85C  EOR #FF     et on fini la complémentation
D7A3  JMP $D775    D85E  JMP $D830   et finir comme LEFT$
```

'MID$' (Fonction)

Principe: comme pour RIGHT$, on calcule la position dans la chaine originale et

la longueur de la chaine résultat, en prenant garde aux divers dépassements.

```
D7A6  LDA #FF      D861  LDA #FF      prendre valeur par défaut
D7A8  STA D4       D863  STA D4       du dernier paramètre
D7AA  JSR $00E8    D865  JSR $00E8    prendre caractère courant
D7AD  CMP #')'     D868  CMP #')'     et tester si ')'
D7AF  BEQ D7B7     D86A  BEQ D872     sauter,on a tous les paramètres
D7B1  JSR $CFD9    D86C  JSR $D065    on veut une ','
D7B4  JSR $D80D    D86F  JSR $D8C8    et un entier dans #D4
D7B7  JSR $D7D0    D872  JSR $D88B    on récupère les autres paramètres
D7BA  BEQ D807     D875  BEQ D8C2    si longueur à prendre=0, erreur
D7BC  DEX          D877  DEX          ajuster position dans la chaine
D7BD  TXA          D878  TXA
D7BE  PHA          D879  PHA          et sauver sur la pile
D7BF  CLC          D87A  CLC
D7C0  LDX #00      D87B  LDX #00      préparer pour chaine vide
D7C2  SBC (BF),Y   D87D  SBC (BF),Y   enlever longueur de la chaine
D7C4  BCS D77C     D87F  BCS D837     si plus courte que paramétre, prendre vide
D7C6  EOR #FF      D881  EOR #FF      sinon,calculer la longueur restante
D7C8  CMP D4       D883  CMP D4       et comparer à longueur demandée
D7CA  BCC D77D     D885  BCC D838     si trop longue,on prend celle restante
D7CC  LDA D4       D887  LDA D4       sinon,prendre la longueur demandée
D7CE  BCS D77D     D889  BCS D838     en tous cas, faire fin de LEFT$
```

RECUPERER LES PARAMETRES POUR LES FONCTIONS CHAINE

```
D7D0  JSR $CFD3    D88B  JSR $D05F    Demande ")"
D7D3  PLA          D88E  PLA          prendre adresse de retour
D7D4  TAY          D88F  TAY          dans Y
D7D5  PLA          D890  PLA
D7D6  STA C4       D891  STA C4       et #C4
D7D8  PLA          D893  PLA          enlever adresse de retour pour éviter
D7D9  PLA          D894  PLA          le test de résultat numérique
D7DA  PLA          D895  PLA          puis sortir le paramètre entier
D7DB  TAX          D896  TAX          dans X
D7DC  PLA          D897  PLA          puis adresse descripteur poids faible
D7DD  STA BF       D898  STA BF       dans #BF
D7DF  PLA          D89A  PLA          et poids fort
D7E0  STA C0       D89B  STA C0       dans #C0
D7E2  LDA C4       D89D  LDA C4       récupérer adresse de retour
D7E4  PHA          D89F  PHA          et la restaurer
D7E5  TYA          D8A0  TYA          sur la pile
D7E6  PHA          D8A1  PHA
```

```
D7E7  LDY #ØØ     D8A2  LDY #ØØ     initialiser Y
D7E9  TXA         D8A4  TXA         et 2 ème paramètre dans A, placer Z.
D7EA  RTS         D8A5  RTS
```

'LEN' (FONCTION)

```
D7EB  JSR $D7F1   D8A6  JSR $D8AC   enlever la réservation, longueur dans Y
D7EE  JMP $D3FD   D8A9  JMP $D4B6   Y --) ACC1
```

ENLEVER RESERVATION ET INDIQUER RESULTAT NUMERIQUE

Entrée: le descripteur de la chaine est dans ACC1

Sortie: la réservation temporaire a été enlevée, A=Y=longueur de la chaine, Z et N positionnés selon la longueur.

Principe: on enlève l'éventuelle réservation temporaire, et on place l'adresse de la chaine en #91-#92. On indique ensuite que le résultat va être numérique pour ne pas créer de TYPE MISMATCH

```
D7F1  JSR $D712   D8AC  JSR $D7CD   prendre descripteur
D7F4  LDX #ØØ     D8AF  LDX #ØØ
D7F6  STX 28      D8B1  STX 28      et indiquer résultat numérique
D7F8  TAY         D8B3  TAY         positionner Z selon la longueur
D7F9  RTS         D8B4  RTS
```

'ASC' (FONCTION)

```
D7FA  JSR $D7F1   D8B5  JSR $D8AC   prendre descripteur
D7FD  BEQ D8Ø7    D8B8  BEQ D8C2    si chaine vide, ILLEGAL QUANTITY
D7FF  LDY #ØØ     D8BA  LDY #ØØ
D8Ø1  LDA (91),Y  D8BC  LDA (91),Y  prendre premier caractère de la chaine
D8Ø3  TAY         D8BE  TAY         dans Y
D8Ø4  JMP $D3FD   D8BF  JMP $D4B6   Y --) ACC1 non signé
D8Ø7  JMP $D2AØ   D8C2  JMP $D336   ILLEGAL QUANTITY
```

SAUTER UN CARACTERE ET PRENDRE UN ENTIER DANS X

```
D8ØA  JSR $ØØE2   D8C5  JSR $ØØE2
```

PRENDRE UN ENTIER DANS X

Entrée: TXTPTR bien placé

Sortie: X contient l'entier évalué
A , Z et C placés selon le caractère suivant l'expression.

```
D80D  JSR $CE77   D8C8  JSR $CF03   évaluer une expression numérique
D810  JSR $D210   D8CB  JSR $D2A2   ACC1 --> #D4-#D3 non signé
D813  LDX D3      D8CE  LDX D3      si poids fort non nul,
D815  BNE D807    D8D0  BNE D8C2    ILLEGAL QUANTITY
D817  LDX D4      D8D2  LDX D4      si OK, prendre poids faible
D819  JMP $00E8   D8D4  JMP $00E8   et positionner selon prochain caractère
```

'VAL' (FONCTION)

Principe: on va lancer l'évaluation normale d'un nombre. Cette routine se basant sur TXTPTR, on va le sauver, puis le placer au début de la chaine à évaluer. Il faut aussi veiller à ce que l'évaluation s'arrête, il faut donc mettre un terminateur (00) à la fin de la chaine, on va donc sauver le caractère, qui peut être celui d'une autre chaine, puis le récupérer après l'évaluation.

```
D81C  JSR $D7F1   D8D7  JSR $D8AC   prendre le descripteur
D81F  BNE D824    D8DA  BNE D8DF
D821  JMP $DB27   D8DC  JMP $DBB2   si chaine vide,ACC1=0
D824  LDX E9      D8DF  LDX E9
D826  LDY EA      D8E1  LDY EA      prendre TXTPTR
D828  STX E0      D8E3  STX E0
D82A  STY E1      D8E5  STY E1      et le sauver
D82C  LDX 91      D8E7  LDX 91      adresse de la chaine poids faible
D82E  STX E9      D8E9  STX E9      dans TXTPTR
D830  CLC         D8EB  CLC         on ajoute la longueur
D831  ADC 91      D8EC  ADC 91
D833  STA 93      D8EE  STA 93      et on sauve
D835  LDX 92      D8F0  LDX 92
D837  STX EA      D8F2  STX EA      poids fort de la chaine dans TXTPTR
D839  BCC D83C    D8F4  BCC D8F7    sauter si pas de report
D83B  INX         D8F6  INX         ajuster poids fort de la fin
D83C  STX 94      D8F7  STX 94      et sauver adresse de fin
D83E  LDY #00     D8F9  LDY #00     indexer début chaine suivante
D840  LDA (93),Y  D8FB  LDA (93),Y  et prendre le premier caractère
```

```
D842  PHA          D8FD  PHA          le sauver
D843  LDA #00      D8FE  LDA #00      et mettre un terminateur (TYA serait mieux)
D845  STA (93),Y   D900  STA (93),Y   pour convertir correctement
D847  JSR $00E8    D902  JSR $00E8    prendre premier caractère de la chaine
D84A  JSR $DFCF    D905  JSR $DFE7    et évaluer un nombre dans ACC1
D84D  PLA          D908  PLA          récupérer le caractère
D84E  LDY #00      D909  LDY #00      ajuster index
D850  STA (93),Y   D90B  STA (93),Y   et le remettre à sa place
```

RECUPERER TXTPTR DANS #E0-#E1

```
D852  LDX E0       D90D  LDX E0
D854  LDY E1       D90F  LDY E1       prendre sauvegarde
D856  STX E9       D911  STX E9
D858  STY EA       D913  STY EA       et remettre dans TXTPTR
D85A  RTS          D915  RTS
```

PRENDRE 2 PARAMETRES ENTIERS

Entrée: TXTPTR correctement placé

Sortie: #33-#34 contient le premier paramètre et X le second, qui doivent être séparés par une virgule.

On sort par un JMP #00E8, donc A, Z, C sont positionnés selon le caractère suivant.

```
D85B  JSR $CE77    D916  JSR $CF03    évaluer une expression numérique
D85E  JSR $D867    D919  JSR $D922    ACC1 --> #D4-#D3 (non signé)
D861  JSR $CFD9    D91C  JSR $D065    demander ','
D864  JMP $D80D    D91F  JMP $D8C8    puis prendre un entier dans X
```

ACC1 --> YA NON SIGNE

Entrée: ACC1 contient la valeur désirée

Sortie: YA, #D4-#D3, #33-#34 contiennent la valeur convertie en entier. Z et N sont positionnés selon le poids faible

```
D867  LDA D5       D922  LDA D5       prendre signe
D869  BMI D807     D924  BMI D8C2     si négatif, 'ILLEGAL QUANTITY'
D86B  LDA D0       D926  LDA D0       prendre exposant
```

```
D86D  CMP #91     D928  CMP #91     si supérieur à 16
D86F  BCS D807    D92A  BCS D8C2    ILLEGAL QUANTITY aussi
D871  JSR $DF74   D92C  JSR $DF8C   ACC1 --> #D4-#D3-#D2-#D1 signé
D874  LDA D3      D92F  LDA D3
D876  LDY D4      D931  LDY D4      récupérer valeur dans YA
D878  STY 33      D933  STY 33
D87A  STA 34      D935  STA 34      et sauver
D87C  RTS         D937  RTS
```

'PEEK' (FONCTION)

```
D87D  LDA 34      D938  LDA 34
D87F  PHA         D93A  PHA
D880  LDA 33      D93B  LDA 33      il faut sauver à cause du POKE
D882  PHA         D93D  PHA
D883  JSR $D867   D93E  JSR $D922   ACC1 --> #33-#34
D886  LDY #00     D941  LDY #00
D888  LDA (33),Y  D943  LDA (33),Y  prendre la valeur
D88A  TAY         D945  TAY         et la sauver dans Y
D88B  PLA         D946  PLA
D88C  STA 33      D947  STA 33      et récupérer #33
D88E  PLA         D949  PLA
D88F  STA 34      D94A  STA 34      et #34
D891  JMP $D3FD   D94C  JMP $D4B6   Y --> ACC1 (non signé)
```

'POKE' (COMMANDE)

Remarque: étant donné sa configuration (on met l'adresse dans #33-#34, puis on va chercher la valeur), il est nécéssaire qu'aucune fonction ne touche à #33-#34, quitte à le sauver provisoirement (Cf PEEK par exemple).

Curieuse complication,il aurait suffit de faire comme pour DOKE: utiliser #1D-#1E.

Bogue: sur la V1.0, POKE avec un nombre hexadécimal donne vraiment n'importe quoi. Ce n'est pas le POKE qui est en cause, ou du moins pas directement. Simplement,la routine d'évaluation d'un nombre hexa utilise #33-#34, qui contient à la fin la valeur à POKER, au lieu de contenir l'adresse.

De sorte que POKE X,Y avec Y en hexa est équivalent à POKE Y,Y , c'est à dire mettre n'importe quoi dans la page 0. Selon où on tombe, ce n'est pas triste.

La bogue a été corrigée en prenant d'autres variables systèmes que #33-#34 (voir #E848/#E981).

```
D894  JSR $D85B   D94F  JSR $D916   prendre adresse dans #33-#34 et valeur ds X
D897  TXA         D952  TXA         valeur dans A
D898  LDY #00     D953  LDY #00     indexer
D89A  STA (33),Y  D955  STA (33),Y  et placer la valeur
D89C  RTS         D957  RTS
```

'WAIT' (COMMANDE)

Principe: on initalise le Timer 2 à la valeur précisée, et on attend qu'il tombe à Ø. Le timer est décrémenté à chaque intérruption, soit tous les 1/1ØØ ème de seconde en temps normal. Bien entendu, si on inhibe les IRQ, WAIT bloque.
...

Programmation: une utilisation plus rationnelle des routine aurait pu donner, pour la V1.1 par exemple:

```
        JSR $E853
        TAX
        LDA #02
        JMP $EEC9
```

```
D89D  JSR $CE77   D958  JSR $CFØ3   évaluer un nombre
D8AØ  JSR $D867   D95B  JSR $D922   ACC1 --) #33-#34
D8A3  LDY 33      D95E  LDY 33
D8A5  LDX 34      D96Ø  LDX 34      prendre le nombre
D8A7  LDA #02     D962  LDA #02     indiquer Timer No 2
D8A9  JMP $EBDC   D964  JMP $EEC9   YX --) timer 2 et attendre Timer 3=Ø
```

'DOKE' (COMMANDE)

Programmation: depuis le WAIT, la #E79D/#E853 a été découverte. Bientot, peut être, YA sera bien positionné. Curieux cette boucle pour placer 2 valeurs: on gagne un octet, après en avoir perdu 6 dans le WAIT...

```
D8AC  JSR $E79D   D967  JSR $E853   évaluer l'adresse
D8AF  LDA 33      D96A  LDA 33
D8B1  LDY 34      D96C  LDY 34      et prendre l'adresse et la sauver
D8B3  STA 1D      D96E  STA 1D      (inutile, elle est dans YA)
D8B5  STY 1E      D97Ø  STY 1E
D8B7  JSR $CFD9   D972  JSR $DØ65   demander ','
D8BA  JSR $E79D   D975  JSR $E853   prendre la valeur à placer
D8BD  LDY #01     D978  LDY #01     préparer pour décaler 2 octets
D8BF  LDA ØØ33,Y  D97A  LDA ØØ33,Y  prendre la valeur
```

```
D8C2  STA (1D),Y  D97D  STA (1D),Y  et la placer
D8C4  DEY         D97F  DEY
D8C5  BPL D8BF    D980  BPL D97A    et suivant...
D8C7  RTS         D982  RTS
```

'DEEK' (FONCTION)

```
D8C8  JSR $D867   D983  JSR $D922   ACC1 --> #33-#34
D8CB  LDY #01     D986  LDY #01     indexer poids fort
D8CD  LDA (33),Y  D988  LDA (33),Y  prendre poids fort du résultat
D8CF  PHA         D98A  PHA         et sauver sur la pile
D8D0  DEY         D98B  DEY         indexer poids faible
D8D1  LDA (33),Y  D98C  LDA (33),Y  et le prendre
D8D3  TAY         D98E  TAY         dans Y
D8D4  PLA         D98F  PLA         YA contient la valeur
.............     D990  JMP $DF40   YA --> ACC1 (non signé)
```

YA --> ACC1 (non signé)

Principe: on convertit d'abord comme un nombre signé, ce qui veut dire que les nombres de 37268 à 65535 seront en fait -32767 à -1, les autres étant corrects. Il suffit d'ajouter 65536 pour retrouver le bon résultat.

```
D8D5  JSR $D3ED   .............  YA --> ACC1 (signé)
D8D8  BIT D5      .............  le résultat est-il positif ?
D8DA  BPL D8E3    .............  oui, c'est bon
D8DC  LDA #E4     .............  non, on ajoute #10000 pour retrouver
D8DE  LDY #D8     .............  la bonne valeur. AY pointe sur 65536
D8E0  JMP $DA97   .............  ACC1=ACC1+(AY)
D8E3  RTS         .............

D8E4 ....  BYT #91,#00,#00,#00,#00  65536 soit #10000
D8E9 ....  BYT #82,#49,#0F,#DA,#9E  3,14159265 soit PI
```

'PI' (FONCTION)

PI est une fonction, constante certes, mais une fonction, et non une pseudo variable comme certains l'ont appellé. Une pseudo variable est traitée par la routine qui recherche l'adresse d'une variable, et qui intercepte si elle reconnait un nom de variable interne (les variables TIME de certains ordinateurs sont traitées ainsi). L'Oric n'a pas de pseudo variable. Certains

ordinateurs traitent effectivement PI comme une pseudo variable.

```
D8EE  LDA #E9     ..............
D8FØ  LDY #D8     .............. indexer la valeur de PI
D8F2  JMP $DE73   .............. (AY) --> ACC1
```

Convertir A en hexa

Principe: très classique, on décompose en deux digits, et on ajoute le code ASCII du Ø, puis on ajoute encore 7 pour trouver les A,B... si nécessaire

```
D8F5  PHA         D993  PHA         sauver le nombre
D8F6  LSR A       D994  LSR A       et amener quartet de poids fort
D8F7  LSR A       D995  LSR A       dans quartet de poids faible
D8F8  LSR A       D996  LSR A       b7 b6 b5 b4 x x x x -> Ø Ø Ø Ø b7 b6 b5 b4
D8F9  LSR A       D997  LSR A
D8FA  JSR $D8FE   D998  JSR $D99C   convertir en ASCII
D8FD  PLA         D99B  PLA         récupérer nombre
```

Convertir en 1 digit ASCII

```
D8FE  AND #ØF     D99C  AND #ØF     et isoler le quartet de poids faible
D9ØØ  ORA #'Ø'    D99E  ORA #'Ø'    ajouter le code du 'Ø' (onc #3Ø-#3F)
D9Ø2  CMP #':'    D9AØ  CMP #':'    est-ce un chiffre ou une lettre ?
D9Ø4  BCC D9Ø8    D9A2  BCC D9A6    c'est un chiffre, c'est bon
D9Ø6  ADC #Ø6     D9A4  ADC #Ø6     +6+C=+7:#41-#46, on a une lettre
D9Ø8  CMP #'Ø'    D9A6  CMP #'Ø'    au fait, est-ce Ø ?
D9ØA  BNE D91Ø    D9A8  BNE D9AE    non, on sauve (et on saute)
D9ØC  LDY 2F      D9AA  LDY 2F      oui, prendre l'indicateur de nombre nul
D9ØE  BEQ D916    D9AC  BEQ D9B4    il l'est, un Ø devant le nombre est inutile
D91Ø  STA 2F      D9AE  STA 2F      on indique nombre non nul
D912  STA Ø1ØØ,X  D9BØ  STA Ø1ØØ,X  et on recopie dans le tampon
D915  INX         D9B3  INX         et on indexe le chiffre suivant
D916  RTS         D9B4  RTS
```

'HEX$' (FONCTION)

Principe: assez simple, la fonction est un peu compliquée par le fait qu'on veut ignorer les 'Ø' devant le nombre si celui-ci a moins de 4 digits significatifs. On utilise l'indictateur #2F qui est nul tant qu'un digit significatif n'a pas été rencontré.

Bogue: sur la V1.Ø, si le nombre est nul, le résultat est tout simplement #, ce qui est laid.

```
D917  JSR $D867    D9B5  JSR $D922    ACC1 --) #33-#34
D91A  LDX #ØØ      D9B8  LDX #ØØ      préparer index
D91C  STX 2F       D9BA  STX 2F       indiquer nombre nul pour l'instant...
D91E  LDA #'#'     D9BC  LDA #'#'     identificateur hexa
D92Ø  STA FF       D9BE  STA FF       au début de la chaine
D922  LDA 34       D9CØ  LDA 34       prendre poids fort
D924  JSR $D8F5    D9C2  JSR $D993    et convertir
D927  LDA 33       D9C5  LDA 33       prendre poids faible
D929  JSR $D8F5    D9C7  JSR $D993    et convertir aussi
.....              D9CA  TXA          le nombre est-il nul ?
.....              D9CB  BNE D9D3     non,on saute
.....              D9CD  LDA #'Ø'     oui,on met un Ø,c'est plus joli
.....              D9CF  STA Ø1ØØ,X   dans le tampon
.....              D9D2  INX          et on ajuste pour mettre le Ø
D92C  LDA #ØØ      D9D3  LDA #ØØ      prendre terminateur
D92E  STA Ø1ØØ,X   D9D5  STA Ø1ØØ,X   et finir la chaine
D931  JMP $D4EØ    D9D8  JMP $D59B    et finir comme STR$
D934  JMP $CFE4    D9DB  JMP $DØ7Ø    'SYNTAX ERROR'
```

'LORES' (COMMANDE)

Utilisation: passer en mode text, et arriver en #D93D/#D9E4 avec dans X le paramètre (Ø ou 1)

```
D937  JSR $E9A9    D9DE  JSR $EC21    faire TEXT
D93A  JSR $D8ØD    D9E1  JSR $D8C8    prendre une valeur dans X
D93D  TXA          D9E4  TXA
D93E  BEQ D946     D9E5  BEQ D9ED     si c'est Ø, on saute
D94Ø  DEX          D9E7  DEX
D941  BNE D934     D9E8  BNE D9DB     si ce n'est pas 1, erreur
D943  LDA #Ø9      D9EA  LDA #Ø9      attribut pour caractère alterné
D945  BYT #2C      D9EC  BYT #2C      sauter caractère suivant
D946  LDA #Ø8      D9ED  LDA #Ø8      attribut pour caractères normaux
D948  LDX #1Ø      D9EF  LDX #1Ø      attribut fond noir
D94A  STX Ø2F8     D9F1  STX Ø2F8     et on sauve
D94D  LDX #1B      D9F4  LDX #1B      pour les 27 lignes
D94F  PHA          D9F6  PHA          sauver attribut
D95Ø  TXA          D9F7  TXA          ligne dans A
D951  JSR $D965    D9F8  JSR $DAØC    calculer adresse de la A ème ligne
D954  LDA Ø2F8     D9FB  LDA Ø2F8     prendre attribut de couleur
```

```
D957  LDY #27     D9FE  LDY #27     pour les 40 caractères de la ligne
D959  STA (1F),Y  DA00  STA (1F),Y  et l'écrire
D95B  DEY         DA02  DEY         jusqu'au début de la ligne
D95C  BNE D959    DA03  BNE DA00
D95E  PLA         DA05  PLA         récupérer attribut caractère
D95F  STA (1F),Y  DA06  STA (1F),Y  dans colonne 0
D961  DEX         DA08  DEX         passer à la ligne suivante
D962  BNE D94F    DA09  BNE D9F6
D964  RTS         DA0B  RTS
```

CALCULER ADRESSE LIGNE A (MODE TEXT)

Entrée: A contient un numéro de ligne

Sortie: #1F-#20=adresse=#BB80+40*A.

X inchangé. Sur V1.0, Y non plus.
A contient aussi le poids faible du résultat, ça peut servir (pas dans la ROM en tous cas !).

Bogue: sur V1.0, on réécrit la même routine de multiplication que #F700. Curieux !

Calculer 40*A

```
D965  PHA         ..............  sauver la ligne
D966  LDA #00     ..............
D968  STA 20      ..............  initialiser poids fort du résultat
D96A  PLA         ..............  on prend la ligne
D96B  STA 1F      ..............  et on sauve pour future addition
D96D  ASL A       ..............
D96E  ROL 20      ..............  *2
D970  ASL A       ..............
D971  ROL 20      ..............  *4
D973  CLC         ..............
D974  ADC 1F      ..............
D976  BCC D97A    ..............
D978  INC 20      ..............  +original=*5
D97A  ASL A       ..............
D97B  ROL 20      ..............  *10
D97D  ASL A       ..............
D97E  ROL 20      ..............  *20
D980  ASL A       ..............
```

```
D981  ROL 2Ø        .................  ‡4Ø
D983  STA 1F        .................  et sauver le résultat
.................   DAØC  JSR $F731   AY=4Ø‡A
.................   DAØF  STY 2Ø      sauver poids fort
D985  CLC           DA11  CLC
D986  ADC #8Ø       DA12  ADC #8Ø     ajouter poids faible de la base de l'écran
D988  PHA           DA14  PHA         et on sauve
D989  STA 1F        DA15  STA 1F      surtout en #1F
D98B  LDA #BB       DA17  LDA #BB     et pareil pour le poids fort
D98D  ADC 2Ø        DA19  ADC 2Ø
D98F  STA 2Ø        DA1B  STA 2Ø
D991  PLA           DA1D  PLA         on reprend le poids faible
D992  RTS           DA1E  RTS
D993  JMP $D8Ø7     DA1F  JMP $D8C2   'ILLEGAL QUANTITY' (chemin détourné)
```

PRENDRE DEUX COORDONNEES (PLOT...)

Entrée: TXTPTR correctement positionné

Sortie: #2F8 et X contiennent la coordonnée horizontale, #1F-#2Ø l'adresse de la ligne précisée.

Bogue: sur la V1.Ø, on ajoute 1 à la coordonnée horizontale, ce qui ne veut rien dire (ajouter 2 aurait permis d'ignorer les colonnes protégées). En fait on dirait que Ça a été fait pour compenser le fait que la comparaison interdisait l'accès à la colonne 39 ?.

```
D996  JSR $DA6B     DA22  JSR $DAF6   interdire mode HIRES
D999  JSR $D8ØD     DA25  JSR $D8C8   prendre une coordonnée dans X
D99C  CPX #27       DA28  CPX #28     et vérifier pas trop long
D99E  BCS D993      DA2A  BCS DA1F     si c'est hors de l'écran, erreur
D9AØ  INX           .................  curieux !!
D9A1  STX Ø2F8      DA2C  STX Ø2F8    et on sauve la coordonnée horizontale
D9A4  JSR $CFD9     DA2F  JSR $DØ65   demander ','
D9A7  JSR $D8ØD     DA32  JSR $D8C8   prendre 2 ème coordonnée
D9AA  CPX #1B       DA35  CPX #1B     et vérifier pas plus de 27 lignes !
D9AC  BCS D993      DA37  BCS DA1F     oui, erreur
D9AE  INX           DA39  INX         +1 car ligne Ø invalide
D9AF  TXA           DA3A  TXA         dans A
D9BØ  JSR $D965     DA3B  JSR $DA8C   pour calculer l'adresse de la ligne
D9B3  RTS           DA3E  RTS
```

'SCRN' (FONCTION)

Principe: limpide...

Bogue: décalé comme le PLOT sur la V1.Ø (voir routine précédente)

```
D9B4  JSR $CFD6    DA3F  JSR $DØ62   demander '('
D9B7  JSR $D996    DA42  JSR $DA22   prendre coordonnées
D9BA  JSR $CFD3    DA45  JSR $DØ5F   demander ')'
D9BD  LDY Ø2F8     DA48  LDY Ø2F8    prendre position sur la ligne
D9CØ  LDA (1F),Y   DA4B  LDA (1F),Y  prendre le caractère à l'écran
D9C2  TAY          DA4D  TAY         dans Y
D9C3  JMP $D3FD    DA4E  JMP $D4B6   Y --) ACC1 (non signé)
```

'PLOT' (COMMANDE)

Principe: très simple. En fait la rapidité d'affichage du PLOT, vient du fait que le rapport caractères affichés/évaluation de variable est très grand: pour afficher 1ØØ caractères par exemple, il faut trois évaluations avec PLOT et... 6ØØ avec POKE !.

Au sens du Basic, le transfert de la chaine est instantané, de sorte que le PLOT, tout comme POKE, DOKE etc, passe tout son temps à évaluer les variables.

Bogue: le décalage de 1 colonne de la V1.Ø vient de la routine #D996/#DA22

```
D9C6  JSR $D996    DA51  JSR $DA22   prendre deux coordonnées
D9C9  JSR $CFD9    DA54  JSR $DØ65   demander ','
D9CC  JSR $CE8B    DA57  JSR $CF17   évaluer expression
D9CF  BIT 28       DA5A  BIT 28      tester si chaine
D9D1  BPL D9FØ     DA5C  BPL DA7B    non, on saute
D9D3  JSR $D715    DA5E  JSR $D7DØ   oui, on enlève la réservation
D9D6  TAX          DA61  TAX         longueur de la chaine dans X
D9D7  CLC          DA62  CLC         et on calcule l'adresse du début
D9D8  LDA Ø2F8     DA63  LDA Ø2F8    c'est à dire index+Base
D9DB  ADC 1F       DA66  ADC 1F
D9DD  BCC D9E1     DA68  BCC DA6C
D9DF  INC 2Ø       DA6A  INC 2Ø      et aussi poids fort
D9E1  STA 1F       DA6C  STA 1F
D9E3  LDY #ØØ      DA6E  LDY #ØØ     préparer index
D9E5  INX          DA7Ø  INX         ajuster le nombre de boucles
D9E6  DEX          DA71  DEX         décompter le nombre de caractères
D9E7  BEQ D9F9     DA72  BEQ DA84    c'est fini, on sort
```

```
D9E9  LDA (91),Y  DA74  LDA (91),Y  transférer la chaine
D9EB  STA (1F),Y  DA76  STA (1F),Y  vers l'écran
D9ED  INY         DA78  INY         on continue
D9EE  BNE D9E6    DA79  BNE DA71    inconditionnel.
D9F0  JSR $D810   DA7B  JSR $D8CB   ACC1 --> X
D9F3  TXA         DA7E  TXA         code dans A
D9F4  LDY 02F8    DA7F  LDY 02F8    prendre index
D9F7  STA (1F),Y  DA82  STA (1F),Y  et sauver le code
D9F9  RTS         DA84  RTS
```

'REPEAT' (COMMANDE)

Principe: REPEAT agit exactement comme GOSUB pour ce qui est d'empiler TXTPTR et le numéro de la ligne, on indique simplement que c'est REPEAT en mettant son token sur la pile.

Comme pour RETURN, l'adresse de retour est empilée, ce qui facilite le UNTIL, mais oblige à sortir par un saut et non un simple RTS.

```
D9FA  BNE DA13    DA85  BNE DA9E    erreur s'il y a des paramètres
D9FC  LDA #03     DA87  LDA #03     demander 6 octets sur la pile
D9FE  JSR $C43B   DA89  JSR $C437
DA01  LDA EA      DA8C  LDA EA
DA03  PHA         DA8E  PHA         sauver TXTPTR poids fort
DA04  LDA E9      DA8F  LDA E9
DA06  PHA         DA91  PHA         et poids faible
DA07  LDA A9      DA92  LDA A9
DA09  PHA         DA94  PHA         numéro de la ligne poids fort
DA0A  LDA A8      DA95  LDA A8
DA0C  PHA         DA97  PHA         et aussi poids faible
DA0D  LDA #8B     DA98  LDA #8B     prendre code de REPEAT
DA0F  PHA         DA9A  PHA         sur la pile
DA10  JMP $C8AD   DA9B  JMP $C8C1   et on reoturne à l'interpréteur
DA13  JMP $CFE4   DA9E  JMP $D070   'SYNTAX ERROR'
```

'PULL' (COMMANDE)

'UNTIL' (COMMANDE)

Principe: les deux commandes sont différenciées grâce à Y qui contient en arrivant le double du token. Voir POP/RETURN

On recherche le premier bloc qu n'est pas FOR (donc on peut sortir

sans crainte d'une boucle par un UNTIL). Ensuite, si c'est PULL, on simule un UNTIL qui sort, c'est à dire un UNTIL avec condition vraie.

Si la condition est vraie, il suffit de dépiler, et il reste au sommet l'adresse de retour à l'interpréteur.

Si la condition est fausse, on recommence. Curieusement, au lieu dans ce cas de sortir par un simple saut à l'interpréteur, on dépile TXTPTR et le numéro de ligne pour les rempiler immédiatement après. Utilité à démontrer...

```
DA16  LDA #FF      DAA1  LDA #FF      mettre adresse de variable bidon
DA18  STA B9       DAA3  STA B9
DA1A  JSR $C3CA    DAA5  JSR $C3C6   et chercher le 1er bloc non FOR
DA1D  TXS          DAA8  TXS         ajuste la pile dessus
DA1E  CMP #8B      DAA9  CMP #8B     au fait,c'est bien en bloc REPEAT ?
DA20  BEQ DA27     DAAB  BEQ DAB2    oui,c'est bon
DA22  LDX #F5      DAAD  LDX #F5     non,'BAD UNTIL ERROR'
DA24  JMP $C485    DAAF  JMP $C47E
DA27  CPY #10      DAB2  CPY #10     PULL=#88:#88*2=#110 donc 10
DA29  BNE DA30     DAB4  BNE DABB    si ce n'est pas PULL,on saute
DA2B  STY D0       DAB6  STY D0      simuler condition vraie (#D0<>0)
DA2D  TYA          DAB8  TYA         placer Z=0 (inutile,Z=1,BEQ aurait suffit)
DA2E  BNE DA36     DAB9  BNE DAC1    et finir comme UNTIL
DA30  JSR $00E8    DABB  JSR $00E8   inutile: traiter UNTIL
DA33  JSR $CE8B    DABE  JSR $CF17   évaluer la condition
DA36  PLA          DAC1  PLA         dépiler le token REPEAT
DA37  LDA D0       DAC2  LDA D0      condition fausse ?
DA39  BEQ DA40     DAC4  BEQ DACB    oui, on recommence
DA3B  PLA          DAC6  PLA         condition fausse: on ajuste la pile
DA3C  PLA          DAC7  PLA
DA3D  PLA          DAC8  PLA
DA3E  PLA          DAC9  PLA
DA3F  RTS          DACA  RTS         et c'est tout.
DA40  PLA          DACB  PLA
DA41  STA A8       DACC  STA A8      récupérer numéro de ligne
DA43  PLA          DACE  PLA
DA44  STA A9       DACF  STA A9      poids fort
DA46  PLA          DAD1  PLA
DA47  STA E9       DAD2  STA E9      TXTPTR poids faible
DA49  PLA          DAD4  PLA         poids fort aussi
DA4A  STA EA       DAD5  STA EA
DA4C  JMP $DA01    DAD7  JMP $DA8C   et on réempile le tout !!!
```

'KEY$' (COMMANDE)

```
DA4F  JSR $E9Ø5    DADA  JSR $EB78   prendre caractère dans le tampon touche
DA52  PHP          DADD  PHP         sauver si valide
DA53  PHA          DADE  PHA         et le caractère lui-même.
DA54  BPL DA59     DADF  BPL DAE4    si pas de touche, longueur=Ø
DA56  LDA #Ø1      DAE1  LDA #Ø1     si touche, longueur=1
DA58  BYT #2C      DAE3  BYT #2C     sauter instruction suivante
DA59  LDA #ØØ      DAE4  LDA #ØØ     longueur=Ø
DA5B  JSR $D4FØ    DAE6  JSR $D5AB   réserver la chaine
DA5E  PLA          DAE9  PLA         récupérer la touche
DA5F  PLP          DAEA  PLP
DA6Ø  BPL DA66     DAEB  BPL DAF1    si pas de touche, on sort
DA62  LDY #ØØ      DAED  LDY #ØØ
DA64  STA (D1),Y   DAEF  STA (D1),Y  sinon, on sauve dans la chaine
DA66  PLA          DAF1  PLA
DA67  PLA          DAF2  PLA         dépiler retour pour éviter test numérique
DA68  JMP $D539    DAF3  JMP $D5F4   sauver descripteur sur la pile
```

INTERDIRE MODE HIRES

```
DA6B  LDA Ø2CØ     DAF6  LDA Ø2CØ    prendre indicateur de mode
DA6E  AND #Ø1      DAF9  AND #Ø1     et isoler le bit HIRES/TEXT
DA7Ø  BEQ DA77     DAFB  BEQ DBØ2    mode text, c'est bon
DA72  LDX #A3      DAFD  LDX #A3     'DISP TYPE MISMATCH ERROR'
DA74  JMP $C485    DAFF  JMP $C47E
DA77  RTS          DBØ2  RTS
DA78  RTS          DBØ3  RTS          si un ne suffit pas !!
```

E) LE CALCUL FLOTTANT

1-Le codage des nombres

Nous allons faire un résumé des différentes formes de codage des nombres employées sur l'Oric.

Les entiers non signés.

C'est la forme la plus usuelle, que l'on emploie tout le temps. Il s'agit tout simplement d'écrire un nombre en binaire (base 2) au lieu de l'écrire en décimal (base 1Ø).

Rappelons l'expression d'un nombre X dans la base B donnée:

$$X=a_0b^0+a_1b^1+a_2b^2+....$$
$$a_i < b$$

Figure Codage A

La condition $a_i<b$ donnant l'unicité de la décomposition .

Ainsi que le nombre 1234 en base 10 peut se décomposer en 1*1000+2*100+3*10+4*1, le nombre 1101 en base 2 peut se décomposer en 1*1000+1*100+0*10+1*1, ce qui donne en convertissant la décomposition en base 10: 1*8+1*4+0*2+1*1. Le binaire de connait donc que deux chiffres, contre 10 pour le décimal. Ces deux chiffres sont notés 0 et 1.

Le binaire est lourd à manipuler: le nombre 1234 décimal nécessite seulement 4 chiffres alors que son expression en binaire en nécessite 19.

Les chiffres binaires sont donc groupés par 8, pour obtenir des octets.

Rappelons aussi que n chiffres dans la base b permettent d'exprimer des nombres compris entre 0 (inclus) et b puissance n (non inclus).

Un octet permet donc d'exprimer des nombres entre 0 et 255 (2^8-1).

En groupant deux octets, on peut exprimer des nombres entre 0 et 65535 (2^16-1). Le nombre s'exprime alors en base 256: X=B*256+A. A est le premier chiffre, B le deuxième. A est appelé le poids faible et B le poids fort (en base 10, on les aurait appelé respectivement les unités et les dizaines). Par convention pour le 6502, on stockera en mémoire le poids faible d'abord.

Pour faciliter la lecture des nombres codés en binaire, on groupe les bits par 4, et on obtient donc la base 16 dite hexadécimale. Il faut donc 16 chiffres, qui sont 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. C'est la notation la mieux adaptée au travail en assembleur. Un octet nécessite deux chiffres (on dit parfois digit) hexadécimaux. 100 (base10)=64 (base 16)=01100100 (base2).

Les nombres entiers signés

On a souvent besoin de travailler avec des nombres négatifs, pour obtenir une soustraction par exemple.

L'idée la plus simple est de se réserver un bit qui indiquera si le nombre est positif (0) ou négatif (1). Pour un nombre sur un octet on prendra b7, pour un nombre sur 2 octets on prendra b15 etc.

Mais il faut faire attention à ce que des opérations simples donne facilement le résultat escompté.

Par exemple, en décimal, (-1)+(1) doit donner 0.

En utilisant seulement b7 comme bit de signe, -1 est codé #81 et 1 est codé #01. #81+#01=#82, soit -2 avec cette convention, pas très réussi !

Il fut donc imaginé la notation en vrai complément à 2, qui résoud tout ces problèmes.

La complémentation est en fait l'opérateur logique NON, qui consiste à inverser tous les bits.

Le complément est déjà très satisfaisant: -1 est codé #FE et 1 est codé #01. #FE+#01 donne #FF, soit -#00=0. Le problème est là: on a deux 0 ! C'est très peu pratique pour les calculs.

Il suffit de décaler de 1 et tout rentre dans l'ordre. Le complément à 2 s'obtient d'abord en complémentant, puis en ajoutant 1 au résultat.

-1 est donc codé #FE+1=#FF. -1+1=#FF+#01=#100=#00 (+retenue). Cette fois, c'est réussi, on n'a qu'un seul 0. Seul inconvénient, le domaine est dissymétrique par rapport au 0. Le plus petit négatif est #80=-128, le plus grand positif est #7F=+127.

Autre problème: celui du dépassement de capacité. 127+1 donne 128, qui est hors du domaine d'un nombre signé sur un octet, et la retenue n'y peut rien: #7F+1=#80 (-128 !). Les concepteurs du 6502 ont tout prévue: l'indicateur V (oVerflow), qui agit comme une retenue pour un nombre signé: C reflétait les propagations au delà de b7, V reflète les propagations au delà de b6.

Dans le commentaire, lorsqu'un nombre est dit signé, cela signifie que son bit de poids le plus fort est un bit de signe, et donc que les nombres négatifs sont notés en complément à 2.

Rappel: soustraire 1 et ajouter -1 revient donc même quant au résultat.

Les variables du BASIC dites entières (suffixe %) sont stockées en complément à 2 sur 2 octets.

Pour la conversion en décimal, la ROM utilise aussi des nombres signés sur 4 octets.

Les nombres 'réels'

Les nombres signés, c'est bien, mais limité !

Pour pouvoir faire des calculs efficaces, il faudrait travailler sur l'ensemble des réels, dont on peut apréhender la structure continue -et infinie- en disant qu'entre deux nombre réels, il y a toujours un autre nombre réel.

En fait l'ensemble des réels est une extension de l'ensemble des rationnels (ensemble des quotients de nombres entiers relatifs).

Nous ne nous étendrons pas sur la nécessité -ni la manière- de construire l'ensemble des réels, mais retenons simplement qu'il a un caractère infini qui est pour le moins difficile à représenter ... surtout par un ordinateur, quel qu'il soit !

Il est bien entendu impossible de représenter un ensemble infini, cela nécessiterait une infinité de cases mémoire. Il va donc falloir se ramener à un ensemble fini de nombres. Pour ceci, on utilise deux types de limitations: la précision et la grandeur des nombres.

Il est d'usage de noter un nombre en notation dite 'scientifique' sous la forme d'une mantisse (comprise entre 0 et 10) et d'un puissance de 10: X=M*10^E.

La grandeur est alors l'exposant. Une limitation sur la grandeur des nombres est donc une limitation sur l'exposant: si on dit que -3< E < 3, le nombre représenté sera (sans compter son signe et le zéro) compris entre 1000 et 0,001.

La limitation en précision dépend uniquement de la précision de la mantisse, c'est à dire le nombre de chiffres significatifs. Si on limite la mantisse à un chiffre, la mantisse vaudra de 0 à 9. Mais attention: la précision dépend de l'exposant. Ainsi, la précision va varier de+-500 à +-.005, selon l'exposant.

Première conséquense: on n'a plus un ensemble continu mais en ensemble discret de nombres. Ainsi, il y a 10 possibilités pour la mantisse et 6 pour l'exposant: l'ensemble est donc réduit à 6*10=60 nombres.

Deuxième conséquence: quelle que soit la taille du nombre, la précision est toujours de un chiffre significatif. C'est ceci que l'on appelle la notion de virgule flottante: la précision relative des nombres est indépendante de leur grandeur. Par opposition à un codage normal (par exemple un nombre non signé sur un octet) où la précision absolue est constante (1), mais où la précision relative varie de 1/2 à 1/512. Un nombre voisin de 100 sera estimé à 1/100ème près, c'est à dire avec deux chiffres significatifs, alors qu'un nombre voisin de 10 sera exprimé à 1/10ème près, c'est à dire un seul chiffre significatif.

Il a donc été choisi une représentation en virgule flottante qui occupe 5 octets. On ne pourra donc exprimer que au maximum 2^40 nombres différents, soit environ 1000 milliards. C'est beaucoup mais rien devant ce que l'on est sensé représenter.

Bien entendu, on travaille en base 2. La mantisse va donc être comprise entre 0 et 1. On peut représenter un nombre X codé en virgule flottante par la convention suivante:

X=M*2^E, si M=0 alors M=1, et si E=0 alors X=0.

La mantisse est stockée sur 4 octets, le bit le plus significatif étant le signe. Attention: les nombres négatifs ne sont pas stockés en complément à 2. La précision est de l'ordre de 2^31 soit environ neufs chiffres (décimaux) significatifs.

L'exposant est stocké sur un seul octet, signé.

Là encore, il a fallu prendre une convention qui simplifie les calculs. Le complément à deux n'est cette fois pas très pratique.

Il faut pouvoir détecter facilement l'overflow (l'exposant devient trop grand). Or de nombreuses intructions du 6502 de touchent pas à V (INC...), ce qui complique déjà les choses.

Il faut une convention pour un nombre nul, pour ne pas avoir à tester toute la mantisse. La convention la plus simple, c'est que si l'exposant=000, le nombre est nul. Mais comment alors représenter un nombre de l'ordre de 2^0 (1) ?

Enfin, la plupart des calculs sur l'exposant reviennent à multiplier par 2 le nombre (incrémenter l'exposant) ou diviser le nombre par deux (décrémenter l'exposant). Il faut que ces opérations s'effectuent simplement, sans être obligé d'ajuster l'exposant.

C'est donc une convention dérivée du complément à 2 qui a été choisie: on exprime l'exposant en complément à 2, et on enlève #7F. Ainsi, -1 sera codé (#FF-#7F)=#80, 10 sera codé (#0A-#7F)=#8B etc... (voir table des conversions en annexe)

Evidemment cette convention est très satisfaisante:

-L'overflow est détecté facilement (le plus grand exposant vaut #FF=126: si on chercher à l'incrémenter, on passe à 0) par détection du zéro ou de la retenue.

-Il suffit d'incrémenter pour passer d'un exposant négatifs à un exposant positif: -1=#80, 0=#81, 1=#82.

-Le zéro de l'exposant n'a pas d'importance: le fait que le nombre soit plus petit ou plus grand que 1 importe peu pour faire une addition par exemple.

-La convention d'un nombre nul est simple: l'exposant vaut 0. L'exposant varie donc de #FF (+126) à #01 (-128).

Pendant les phases de calculs, le signe du nombre est stocké à part pour ne pas interférer.

Pour avoir un précision suffisante sur la mantisse, il est nécessaire d'avoir un octet suplémentaire qui sert ensuite à faire l'arrondi. Cet octet est calculé lors des additions, divisions etc... ensuite, selon sa valeur (il suffit de tester b7), on va arrondir la mantisse en l'incrémentant ou non.

Cet octet suplémentaire (noté octet ou bit -lorsque seul b7 est important- d'extension de ACC1) suffit à obtenir un précision maximale pour les additions. Il ne suffit pas en revanche pour les multiplications ou additions, d'où une précision réduite à environ 8 chiffres (décimaux) pour la multiplication dans les cas particuliers.

Toujours au sujet de la précision, il est important de noter que la mantisse doit être justifiée, c'est à dire que son bit le plus significatif doit impérativement être b30.

Sinon, cela revient à coder la mantisse sur moins de 30 bits, c'est à dire perdre en précision. Exemple en base 10: 1,0 10^-1 a deux chiffres significatifs, alors 0,1 a un seul chiffre significatifs.

Voici un résumé et un exemple de codage en virgule flottante:

FIGURE CODAGE-B

L'exposant sera noté E et sa valeur correspondante, après codage, e .

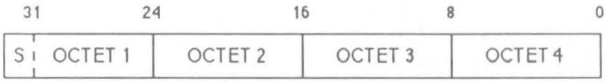
e est calculé en complément à 2, auquel on retranche 127 (ou on ajoute 129) .
e = E - 127 ou e = E+129, soit encore E = e + 127
exemple: si E=15, e=15+129= 144 = #90
ou e= 15-127 = -112 = #90 en complément à 2

- Le nombre est nul si et seulement si e=0 (E= 127)

- Domaine de validité de l'exposant:
  1 < E < 255
  - 128 < E < +126 (domaine dissymétrique)

La MANTISSE , qui représente la valeur du nombre, a deux représentations:
1- pendant les calculs

2- pour le stockage du nombre



S=0: positif
S=1: négatif

Le poids de la mantisse est donc de l'ordre de 2^{E-31}, celui de l'exposant de l'ordre de 2^{E}

Exemple de codage pour le nombre 12,5
$12,5 = 8 + 4 + 1/2 = 2^3 + 2^2 + 2^{-1}$
Exposant = #84 (+3)
Mantisse = 01001000 00000000 00000000 00000000

Figure Codage B

Exposant

Il ne faut jamais perdre de vue que l'Oric travaille sur un ensemble discret de nombres, et que toute opération entraine une perte de précision. Il faut donc simplifier les expressions: A#(B/A) a de fortes chances de ne pas donner B , à cause de l'arrondi de B/A.

Le problème des arrondis semble avoir été optimisé, si on en juge par le travail intense sur l'octet d'extension. Il est aussi assez difficile à suivre, certaines opérations doivent être le fruit de longs calculs de précisions, que j'avoue ignorer.

En résumé, le codage des nombres 'réels' est très satisfaisant, mais ne peut donner plus que 8 chiffres significatifs, et encore au prix de pas mal d'acrobaties. Les routines de calculs sont relativement facile à suivre, si l'on a bien saisi le codage. Elles font en tous cas preuve d'une optimisation à toute épreuve !

2-Approximation des fonctions transcendantes

Ce chapitre est totalement théorique, mais est nécéssaire pour ceux qui voudraient approfondir l'évalutation des fonctions telle que EXP, COS etc..

On ne sait pas calculer exactement la valeur d'une fonction transcendante. Il faut donc trouver la meilleure approximation (la plus rapide à obtenir) compte tenu de l'erreur tolérée.

Le problème posé est le suivant: calculer avec la précision souhaitée (9 chiffres décimaux), et ce le plus rapidement possible, une fonction avec des opérateurs non transcendants tels que +,-,x,/.

Plusieurs méthodes existent. Les deux les plus performantes sont l'approximation par des polynomes, et l'approximation par des fractions continues. La première solution a été retenue, probablement parce que la division est une opération très lente.

Donc, il va falloir trouver un polynome de degré le plus faible possible, pour limiter le temps de calcul et la place nécessaire pour stocker les coefficients, qui doit donner le meilleur résultat, c'est à dire une erreur absolue minimale sur un intervalle donné.

Pour obtenir la valeur d'une fonction au voisinage d'un point, on a recourt à ce qu'on appelle un développement limité, dont voici la définition (formule de TAYLOR):

Soit une fonction f n fois dérivable et de dérivée continue en a, on a alors:

Pour x au voisinage de a:

$$f(x) = \sum_{i=0}^{n} \frac{(x-a)^i}{n!} f^{(n)}(a)$$

$f^{(n)}$ représentant la n-ème dérivée de f

Figure VIE)-A

En clair, on convertit une fonction quelconque en une série polynomiale infinie.

On pourrait croire au premier abord que c'est la solution. En un sens, c'est vrai: en augmentant l'ordre du développement, on peut obtenir n'importe quelle précision sur n'importe quel intervalle (où la fonction est définie et où la série converge. La plupart des D.L ont un rayon de convergence infini).

Mais la définition contient ses limites: c'est une série infinie, chaque terme ajoutant une précision suplémentaire au résultat. Et il est inconcevable de calculer une infinité de termes, évidemment.

D'autre part, un développement limité est en fait une étude locale de la fonction, et n'est valable "qu'au voisinage de " c'est à dire infiniment près du point a où la formule est calculée. Dans cette optique, c'est le moyen le plus performant: la méthode la plus employée pour calculer PI est un D.L de ARCTAN(1)

Plus on s'éloigne du point de calcul, moins le développement est précis, ou plus il faut de termes pour la même précision.

Or, pour les ordinateurs, il faut un résultat précis sur un intervalle assez large (de Ø à 1 par exemple), et on est loin de l'écart 'infiniment petit'. (D'ailleurs, n'importe quelle valeur est infiniment grande par rapport à l'infiniment petit)

Voici un exemple: nous allons étudier la fonction EXP(X), qui est la plus pratique car elle présente la caractéristique d'être égale à sa dérivée. Nous allons l'étudier en Ø car là aussi elle est simple à calculer: EXP(Ø)=1.

La formule du développement limité donne:

Figure VIE)-B

$$e^X = 1 + x + x^2/6 + x^4/24 + x^5/25$$

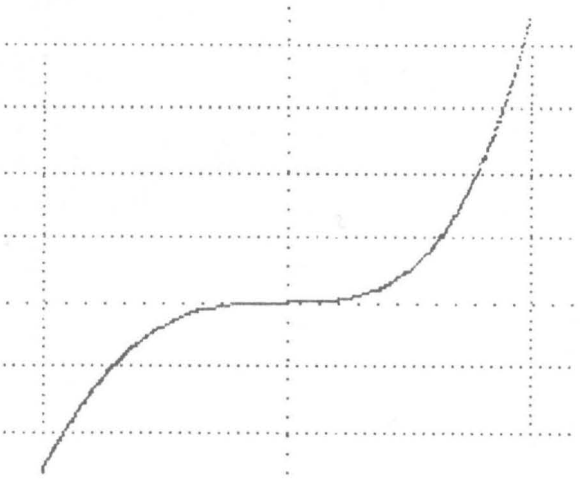
et en notant D_2 le développement tronqué à l'ordre 2

$$D_2(x) = 1 + x + x^2/2$$

Par la suite nous nous limiterons au degré deux, pour ne pas alourdir les calculs.

Traçons la courbe d'erreur de ce D.L (Développement Limité), c'est à dire la fonction EXP(X)-1-X-X^2/2. (l'Oric fournissant 9 chiffres significatifs, la valeur fournie peut être considérée comme exacte devant un D.L à l'ordre 2):

Figure VIE)-C



On le voit, le D.L est d'autant moins précis qu'on s'éloigne de son point de calcul. Pour obtenir en 1 la même précision qu'en Ø,25 par exemple, il faudrait augmenter l'ordre du D.L, c'est à dire ralentir notablement l'exécution, et réserver plus de place pour stocker les coéfficients.

Il a fallu recourrir à d'autre polynomes qui ont une caractéristique plus favorable dans notre cas: la courbe d'erreur est relativement plate sur un intervalle assez large.

On obtient ces polynomes grâce à une 'cuisine' entre la formule de TAYLOR et les polynomes de Chebyshev.

Voici la définition de polynomes de Chebyshev:

Figure VIE)-D

$T_0(x) = 1$, $T_n(x) = \cos(n\vartheta)$, $\cos(\vartheta) = x$

D'où la formule de récurence :

$T_{n+1} = 2 x T_n - T_{(n-1)}$

Voici la liste des premiers polynomes de chebyshev :

$T_0 = 1$

$T_1 = x$

$T_2 = 2x^2 - 1$

$T_3 = 4x^3 - 3x$

$T_4 = 8x^4 - 8x^2 + 1$

$T_5 = 16x^5 - 20x^3 + 5x$

Le tableau suivant est aussi très pratique pour la suite,il exprime cette sois les puissances successives de X en fonction des polynomes de Chebyshev:

Figure VIE)-E

$1 = T0$
$x = T1$
$x^2 = (T0+T2)/2$
$x^3 = (3T1+T3)/4$
$x^4 = (3T0+4T2+T4)/8$
$x^5 = (10T1+5T3+T5)/16$ (1)

Pour calculer la suite des termes, il suffit de noter que la formule de récurence $T_{n+1}=2xT_n - T_{n-1}$ peut s'écrire $xT_n = (T_{n+1} + T_{n-1})/2$
Il suffit de multiplier par x les deux termes de l'égalité (1) par exemple et de remplacer.

Nous ne nous étendrons pas sur les propriétés caractéristiques des polynomes de Chebyshev, nous nous bornerons à constater leur effet.

Voyons donc comment ils vont nous aider à approximer des fonctions. La méthode utilisée est appelée méthode d'économie, et a été créée par Lanczos.

Il faut:

1/ Ecrire un développement limité tronqué à l'ordre n

2/ Convertir en polynomes de Chebyshev grace aux équivalences puissances de X/polynomes de Chebyshev.

3/ Tronquer ce polynome en supprimant un ou deux termes

4/ Retrouver un polynome en X, pour faciliter le calcul.

L'effet est le suivant: les coefficients du D.L vont être modifiés et ce d'autant plus que leur poids est faible (puissances élevées).

Voici une illustration de la méthode, en calculant un D.L à l'ordre 3, puis en tronquant le dernier terme.

$$e^X = 1 + x + \frac{x^2}{2} + \frac{x^3}{6}$$

Figure VIE)-F

$$= T_0 + T_1 + \frac{(T_0 + T_2)}{4} + \frac{(3T_1 + T_3)}{24}$$

$$= \frac{(30\,T_0 + 27\,T_1 + 6\,T_2 + T_3)}{24}$$

Ce qui donne après troncature (il est bien entendu inutile de calculer le terme en T_3)

$$= \frac{(24 + 27x + 12x^2)}{24}$$

$$L(x) = 1 + 1{,}125\,x + 0{,}5\,x^2$$

Essayons de calculer un D.L à l'ordre 4 et 5, en tronquant respectivement les 2 et 3 derniers termes .

$$L(x) = \frac{191}{192} + \frac{25}{24}x + \frac{13}{24}x^2$$

$$L(x) = \frac{191}{192} + \frac{2171}{1920}x + \frac{13}{24}x^2$$

Laquelle de ces approximations est la meilleure ? Est-il nécessaire de calculer des degrés élevés pour tronquer les trois ou quatres derniers termes ?. Gageons que les gens de chez Microsoft ont fait de savants calculs...ou beaucoup d'expériences !. Nous allons voir l'effet pratique des différentes approximations.

Voici les courbes comparées des erreurs absolues:

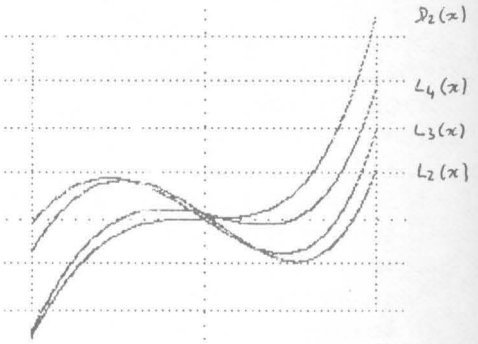


Figure VIE)-G

Ce diagramme permet de déterminer les erreurs maximales. On peut déjà apercevoir la caractéristique des transformées par Chebyshev: jamais très exactes, mais surtout jamais très inexactes.

Pour mieux fixer ces notions, voyons les courbes d'erreur moyenne (Calcul de la valeur moyenne de l'intégrale de la valeur absolue de la différence, ouf !).

L'ordonnée représente l'erreur, et l'abscisse représente la demi largeur d'un intervalle centré sur Ø (1 signifie l'érreur moyenne pour X entre -1 et 1).

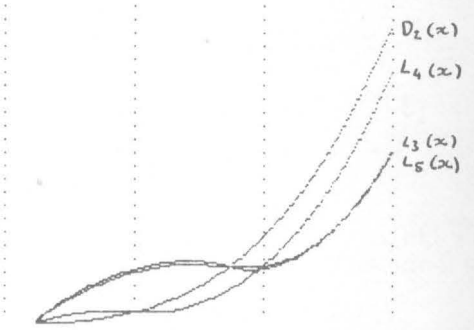


Figure VIE)-H

Les polynomes de Chebyshev montrent toutes leurs vertus: leur erreur moyenne est pratiquement constante entre 1/2 et 1,1. On voit tout de suite qu'il est inutile de calculer des degrés élevés, puisque L3 et L5 ont des courbes d'erreurs quasi identiques.

Le choix entre tous ces polynomes n'est pas simple, et dépend notamment de l'intervalle nécessaire.

En fait, il faut surtout minimiser l'erreur absolue: il serait impensable d'avoir 'en moyenne' 10 chiffres significatifs, si pour certaines valeurs on n'en a plus que 6 ou 7 !

Dans le cas qui nous préoccuppe, il faut prendre en considération l'aspect 'subjectif': EXP(Ø) doit donner 1, c'est à dire que la courbe d'erreur doit s'annuler en Ø, ce qui entraine dans ce cas que le terme constant ne doit pas être modifié, c'est à dire que l'on ne doit tronquer qu'un terme au développement de Chebyshev, et encore si celui-ci est de degré impair.

Les procedés de modifications des D.L. exposés ici ne font pas le tour de la question. D'autres procédés sont utilisés, pour le calcul de LOG par exemple.

Le degré du polynome va dépendre de la rapidité de convergence de la fonction, et aussi de l'intervalle de calcul. Ces détails sont donnés pour chaque fonction, ainsi que les D.L particuliers.

Pour chaque fonction, le problème va donc être de réduire l'intervalle pour pouvoir appliquer un D.L. Cet intervalle devra être -1,1 ou Ø,1 car les polynomes de Chebyshev n'ont des propriétes utiles dans notre cas que dans cet intervalle. Les courbes d'erreur moyenne le montrent bien.

Réduire l'intervalle, quand c'est possible, n'améliore pas toujours la précision car il ne faut pas perdre de vue que l'on travaille sur un ensemble discret de points, et non vraiment sur des réels. Cela tendrait à augmenter l'erreur pour les grands nombres.

C'est ce qu'on appelle l'effet d'arrondi. Un exemple simple:

Supposons que l'on travaille sur un autre ensemble discret, les nombres de Ø à 1Ø par exemple. La formule 2*(9/2) donne 8 au lieu de 9, car 9/2 donne 4. Intuitivement, on peut décrire ce phénomène par une 'perte de résolution'.

Bref,l'approximation des fonctions est un vrai casse tête, fait de nombreux compromis.

ACC1+Ø.5 --) ACC1

```
DA79  LDA #Ø1     DBØ4  LDA #Ø5
DA7B  LDY #E2     DBØ6  LDY #E2     indexer la valeur Ø.5
DA7D  JMP $DA97   DBØ8  JMP $DB22   (AY)+ACC1 --) ACC1
```

(AY)-ACC1 --) ACC1

```
DA8Ø  JSR $DD4D   DBØB  JSR $DD51   (AY) --) ACC2
```

'-' (OPERATEUR) ACC2-ACC1 --) ACC1

Principe: on se ramène à additionner l'opposé...

```
DA83  LDA D5      DBØE  LDA D5      prendre signe ACC1
```

```
DA85  EOR #FF      DB10  EOR #FF      et inverser
DA87  STA D5       DB12  STA D5       sauver nouveau signe
DA89  EOR DD       DB14  EOR DD       et ajuster produit des signes
DA8B  STA DE       DB16  STA DE
DA8D  LDA D8       DB18  LDA D8       positionner N, Z et A selon exposant
DA8F  JMP $DA9A    DB1A  JMP $DB25    et faire '+'

DA92  JSR $DBFB    DB1D  JSR $DC54    justifier la mantisse selon A
DA95  BCC DAD3     DB20  BCC DB5E     inconditionnel
```

(AY)+ACC1 --> ACC1

```
DA97  JSR $DD4D    DB22  JSR $DD51    (AY) --> ACC2
```

'+' (OPERATEUR) ACC2+ACC1 --> ACC1

Entrée: Z doit être positionné selon l'exposant de ACC1 (si tel n'est pas le cas, on risque de faire une addition pour rien si ACC1=0, ce qui n'est pas très grave il est vrai) .

Sortie: résultat dans ACC1, ou par ..OVERFLOW ERROR.

Principe: le premier travail consiste à ajuster les mantisses, de sorte que les nombres aient le même exposant, on va donc décaler le plus petit nombre à droite autant que nécéssaire. Cela revient à 'poser' l'addition.

Ensuite, si les deux nombres sont de même signes, on va simplement additionner les mantisses. S'ils sont de signes opposés, on va faire la soustraction des mantisses, dans le sens correct. Il faut dans ce cas recaler la mantisse à gauche, pour éviter une perte de précision.

```
DA9A  BNE DA9F     DB25  BNE DB2A
DA9C  JMP $DECD    DB27  JMP $DED5    si ACC1=0, ACC2 --> ACC1 et c'est tout
DA9F  LDX DF       DB2A  LDX DF
DAA1  STX C5       DB2C  STX C5       sauver bit d'extension
DAA3  LDX #D8      DB2E  LDX #D6      indexer ACC2
DAA5  LDA D8       DB30  LDA D8       prendre exposant ACC2
DAA7  TAY          DB32  TAY          dans Y
DAA8  BEQ DA78     DB33  BEQ DB03     si nul, le résultat est déjà dans ACC1
DAAA  SEC          DB35  SEC          calculer la différence des exposants
DAAB  SBC D0       DB36  SBC D0
DAAD  BEQ DAD3     DB38  BEQ DB5E     si égaux, sauter
DAAF  BCC DAC3     DB3A  BCC DB4E     si ACC2 < ACC1, on saute
DAB1  STY D0       DB3C  STY D0       sauver exposant ACC2, qui est la référence
```

```
DAB3  LDY DD      DB3E  LDY DD      prendre signe de ACC2
DAB5  STY D5      DB40  STY D5      comme signe du plus grand
DAB7  EOR #FF     DB42  EOR #FF     retrouver la différence des exposants
DAB9  ADC #00     DB44  ADC #00     (C=1), c'est à dire complémenter
DABB  LDY #00     DB46  LDY #00
DABD  STY C5      DB48  STY C5      extension=0 car ACC2 n'a pas d'extension
DABF  LDX #D0     DB4A  LDX #D0     indexer ACC1
DAC1  BNE DAC7    DB4C  BNE DB52    inconditionnel: justifier ACC1 selon A

DAC3  LDY #00     DB4E  LDY #00     ACC2 < ACC1: ACC1 est la référence
DAC5  STY DF      DB50  STY DF      pas d'extension, puis justifier ACC2

Justifier Accu indexé par X, tant que l'exposant <> #D0 (référence)

DAC7  CMP #F9     DB52  CMP #F9     y-a-t-il plus de 8 décalages ?
DAC9  BMI DA92    DB54  BMI DB1D     oui, décaler d'abord octet par octet
DACB  TAY         DB56  TAY         placer index de décalage
DACC  LDA DF      DB57  LDA DF      prendre l'extension
DACE  LSR 01,X    DB59  LSR 01,X    il entre un 0 dans la mantisse
DAD0  JSR $DC14   DB5B  JSR $DC6B   et faire les décalages nécéssaires
DAD3  BIT DE      DB5E  BIT DE      tester produit des signes
DAD5  BPL DB2E    DB60  BPL DBB9     opérandes de même signe, addition

Faire soustraction des mantisses

DAD7  LDY #D0     DB62  LDY #D0     indexer ACC1, plus grand pour l'instant
DAD9  CPX #D8     DB64  CPX #D8     on a justifié ACC2 ?
DADB  BEQ DADF    DB66  BEQ DB6A    oui, c'était bien ACC2 le plus petit
DADD  LDY #D8     DB68  LDY #D8     non, alors le plus grand c'est ACC2
DADF  SEC         DB6A  SEC         préparer pour complémentation
DAE0  EOR #FF     DB6B  EOR #FF     de l'extension
DAE2  ADC C5      DB6D  ADC C5      et commencer par soustraire les extensions
DAE4  STA DF      DB6F  STA DF

DAE6  LDA 0004,Y  DB71  LDA 0004,Y
DAE9  SBC 04,X    DB74  SBC 04,X
DAEB  STA D4      DB76  STA D4      soustraire Octet 4
DAED  LDA 0003,Y  DB78  LDA 0003,Y
DAF0  SBC 03,X    DB7B  SBC 03,X
DAF2  STA D3      DB7D  STA D3      soustraire Octet 3
DAF4  LDA 0002,Y  DB7F  LDA 0002,Y
DAF7  SBC 02,X    DB82  SBC 02,X
DAF9  STA D2      DB84  STA D2      soustraire octet 2
DAFB  LDA 0001,Y  DB86  LDA 0001,Y
```

```
DAFE  SBC Ø1,X    DB89  SBC Ø1,X
DBØØ  STA D1      DB8B  STA D1      soustraire Octet 1
```

JUSTIFIER LA MANTISSE SELON SIGNE

Entrée: C=1 si signe positif, C=Ø sinon. ¤DØ contient l'exposant de départ

Sortie: ACC1 contient le nombre en flottant.

```
DBØ2  BCS DBØ7    DB8D  BCS DB92    si positif demandé, on saute
DBØ4  JSR $DBA9   DB8F  JSR $DCØ2   sinon, il faut complémenter
```

JUSTIFIER LA MANTISSE

Principe: pour avoir une précision constante, il est nécéssaire que la mantisse soit justifiée, c'est à dire que son bit de poids le plus fort sois bien un 1.

Il faut faire une telle justification après une soustraction par exemple, qui a probablement fait diminuer la mantisse.

```
DBØ7  LDY ¤ØØ     DB92  LDY ¤ØØ     extension=Ø si nécessaire
DBØ9  TYA         DB94  TYA         nombre de décalages=Ø
DBØA  CLC         DB95  CLC         préparer C pour additions futures
DBØB  LDX D1      DB96  LDX D1      si poids fort mantisse non nul,
DBØD  BNE DB59    DB98  BNE DBE4    décaler bit à bit
```

Décaler octet par octet

```
DBØF  LDX D2      DB9A  LDX D2      octet 2
DB11  STX D1      DB9C  STX D1      dans octet 1
DB13  LDX D3      DB9E  LDX D3      octet 3
DB15  STX D2      DBAØ  STX D2      dans octet 2
DB17  LDX D4      DBA2  LDX D4      octet 4
DB19  STX D3      DBA4  STX D3      dans octet 3
DB1B  LDX DF      DBA6  LDX DF      octet d'extension
DB1D  STX D4      DBA8  STX D4      dans octet 4
DB1F  STY DF      DBAA  STY DF      nouvelle extension
DB21  ADC ¤Ø8     DBAC  ADC ¤Ø8     exposant+8
DB23  CMP ¤28     DBAE  CMP ¤28     a-t-on décalé 5 fois ?
DB25  BNE DBØB    DBBØ  BNE DB96    non, on continue
```

```
                                ACC1=Ø

DB27  LDA #ØØ     DBB2  LDA #ØØ     oui, la mantisse était vide,et résultat nul
DB29  STA DØ      DBB4  STA DØ      indiquer résultat nul
DB2B  STA D5      DBB6  STA D5      et signe positif.
DB2D  RTS         DBB8  RTS

                Addition (suite):additionner les mantisses

DB2E  ADC C5      DBB9  ADC C5
DB3Ø  STA DF      DBBB  STA DF      additionner extension d'abord
DB32  LDA D4      DBBD  LDA D4
DB34  ADC DC      DBBF  ADC DC
DB36  STA D4      DBC1  STA D4      octet 4
DB38  LDA D3      DBC3  LDA D3
DB3A  ADC DB      DBC5  ADC DB
DB3C  STA D3      DBC7  STA D3      octet 3
DB3E  LDA D2      DBC9  LDA D2
DB4Ø  ADC DA      DBCB  ADC DA
DB42  STA D2      DBCD  STA D2      octet 2
DB44  LDA D1      DBCF  LDA D1
DB46  ADC D9      DBD1  ADC D9
DB48  STA D1      DBD3  STA D1      octet 1
DB4A  JMP $DB66   DBD5  JMP $DBF1   et ajuster exposant


Justification bit à bit

DB4D  ADC #Ø1     DBD8  ADC #Ø1     incrémenter compteur de décalages
DB4F  ASL DF      DBDA  ASL DF      rotation extension
DB51  ROL D4      DBDC  ROL D4      octet 4
DB53  ROL D3      DBDE  ROL D3      octet 3
DB55  ROL D2      DBEØ  ROL D2      octet 2
DB57  ROL D1      DBE2  ROL D1      octet 1
DB59  BPL DB4D    DBE4  BPL DBD8   si toujours pas justifié, on continue
DB5B  SEC         DBE6  SEC         sinon,on ajuste l'exposant
DB5C  SBC DØ      DBE7  SBC DØ      si on a fait plus de décalage
DB5E  BCS DB27    DBE9  BCS DBE2   que l'exposant le permettait, ACC1=Ø
DB6Ø  EOR #FF     DBEB  EOR #FF     complémenter pour retrouver exposant
DB62  ADC #Ø1     DBED  ADC #Ø1
DB64  STA DØ      DBEF  STA DØ      et sauver nouvel exposant
DB66  BCC DBA8    DBF1  BCC DCØ1
DB68  INC DØ      DBF3  INC DØ      incrémenter exposant
DB6A  BEQ DBEØ    DBF5  BEQ DC39    'OVERFLOW ERROR' si il passe à Ø
```

Décaler la mantisse d'un bit à droite

Programmation: Sur la V1.Ø, l'instruction ROR était manifestement inconnue, ou du moins ne fut pas utiliser pour des raisons obscures. La différence de vitesse entre les deux ROM's s'explique par cette routine (procédé aussi utilisé en #DCØ8).

```
DB6C  LDA #ØØ     .................
DB6E  BCC DB72    ................. l'instruction ROR ayant
DB7Ø  LDA #8Ø     ................. disparu depuis une dizaine d'octets
DB72  LSR D1      ................. il a fallu faire sans !
DB74  ORA D1      .................
DB76  STA D1      ................. résultat:la routine nécéssite
DB78  LDA #ØØ     ................. entre 7Ø et 75 microsecondes (selon valeur)
DB7A  BCC DB7E    ................. pour s'exécuter,sans compter
DB7C  LDA #8Ø     ................. que l'on perd 5Ø octets !
DB7E  LSR D2      .................
DB8Ø  ORA D2      .................
DB82  STA D2      .................
DB84  LDA #ØØ     .................
DB86  BCC DB8A    .................
DB88  LDA #8Ø     .................
DB8A  LSR D3      .................
DB8C  ORA D3      .................
DB8E  STA D3      .................
DB9Ø  LDA #ØØ     .................
DB92  BCC DB96    .................
DB94  LDA #8Ø     .................
DB96  LSR D4      .................
DB98  ORA D4      .................
DB9A  STA D4      .................
DB9C  LDA #ØØ     .................
DB9E  BCC DBA2    .................
DBAØ  LDA #8Ø     .................
DBA2  LSR DF      .................
DBA4  ORA DF      .................
DBA6  STA DF      .................
.................  DBF7  ROR D1
.................  DBF9  ROR D2      temps d'exécution:2Ø microsecondes.
.................  DBFB  ROR D3
.................  DBFD  ROR D4
DBA8  RTS          DCØ1  RTS
```

COMPLEMENTER LA MANTISSE

Principe: classique:on inverse chacun des bits et on ajoute 1.

Remarque: il n'y a pas de propagation vers l'exposant,car cette routine n'est appelée qu'à bon escient.

```
DBA9  LDA D5      DC02  LDA D5
DBAB  EOR #FF     DC04  EOR #FF
DBAD  STA D5      DC06  STA D5      changer le signe
DBAF  LDA D1      DC08  LDA D1
DBB1  EOR #FF     DC0A  EOR #FF
DBB3  STA D1      DC0C  STA D1      complémenter octet 1
DBB5  LDA D2      DC0E  LDA D2
DBB7  EOR #FF     DC10  EOR #FF
DBB9  STA D2      DC12  STA D2      complémenter octet 2
DBBB  LDA D3      DC14  LDA D3
DBBD  EOR #FF     DC16  EOR #FF
DBBF  STA D3      DC18  STA D3      complémnter octet 3
DBC1  LDA D4      DC1A  LDA D4
DBC3  EOR #FF     DC1C  EOR #FF
DBC5  STA D4      DC1E  STA D4      complémenter octet 4
DBC7  LDA DF      DC20  LDA DF
DBC9  EOR #FF     DC22  EOR #FF
DBCB  STA DF      DC24  STA DF      complémenter extension
DBCD  INC DF      DC26  INC DF      incrémenter l'extension
DBCF  BNE DBDF    DC28  BNE DC38
DBD1  INC D4      DC2A  INC D4      éventuellement octet 4
DBD3  BNE DBDF    DC2C  BNE DC38
DBD5  INC D3      DC2E  INC D3      éventuellement octet 3
DBD7  BNE DBDF    DC30  BNE DC38
DBD9  INC D2      DC32  INC D2      éventullement octet 2
DBDB  BNE DBDF    DC34  BNE DC38
DBDD  INC D1      DC36  INC D1      éventuellement octet 1
DBDF  RTS         DC38  RTS

DBE0  LDX #45     DC39  LDX #45
DBE2  JMP $C485   DC3B  JMP $C47E   'OVERFLOW ERROR'
```

JUSTIFIER A DROITE ACC3

```
DBE5  LDX #94     DC3E  LDX #94     indexr ACC3
```

JUSTIFIER A DROITE SELON X ET A

Entrée: X contient l'index de l'accumulateur concerné.
A contient 'l'exposant' de départ. On décale à droite tant que A n'est pas nul.

Remarque: Divers points d'entrée...

Décaler octet par octet

```
DBE7  LDY Ø4,X    DC4Ø  LDY Ø4,X    octet 4
DBE9  STY DF      DC42  STY DF      dans extension
DBEB  LDY Ø3,X    DC44  LDY Ø3,X    octet 3
DBED  STY Ø4,X    DC46  STY Ø4,X    dans octet 4
DBEF  LDY Ø2,X    DC48  LDY Ø2,X    octet 3
DBF1  STY Ø3,X    DC4A  STY Ø3,X    dans octet 3
DBF3  LDY Ø1,X    DC4C  LDY Ø1,X    octet 1
DBF5  STY Ø2,X    DC4E  STY Ø2,X    dans octet 2
DBF7  LDY D7      DC5Ø  LDY D7      code de remplissage
DBF9  STY Ø1,X    DC52  STY Ø1,X    dans octet 1

DBFB  ADC #Ø8     DC54  ADC #Ø8     ajouter 8 à 'l'exposant'
DBFD  BMI DBE7    DC56  BMI DC4Ø    pas encore positif: décaler octet par octet
DBFF  BEQ DBE7    DC58  BEQ DC4Ø    si nul, pareil
DCØ1  SBC #Ø8     DC5A  SBC #Ø8     si dépassement, on retrouve la valeur
DCØ3  TAY         DC5C  TAY         comme index
DCØ4  LDA DF      DC5D  LDA DF      l'extension dans A
DCØ6  BCS DC44    DC5F  BCS DC75    si pas de décalage (A=Ø),on sort
DCØ8  PHA         ..............
DCØ9  LDA Ø1,X    ..............
DCØB  AND #8Ø     ..............
DCØD  LSR Ø1,X    ..............    toujours les mêmes instructions compliquées
DCØF  ORA Ø1,X    ..............    et lentes...
DC11  STA Ø1,X    ..............
DC13  BIT 48      ..............
DC15  LDA #ØØ     ..............
DC17  BCC DC1B    ..............
DC19  LDA #8Ø     ..............
DC1B  LSR Ø2,X    ..............
DC1D  ORA Ø2,X    ..............
DC1F  STA Ø2,X    ..............
DC21  LDA #ØØ     ..............
DC23  BCC DC27    ..............
DC25  LDA #8Ø     ..............
```

```
DC27  LSR Ø3,X    .................
DC29  ORA Ø3,X    .................
DC2B  STA Ø3,X    .................
DC2D  LDA #ØØ     .................
DC2F  BCC DC33    .................
DC31  LDA #8Ø     .................
DC33  LSR Ø4,X    .................
DC35  ORA Ø4,X    .................
DC37  STA Ø4,X    .................
DC39  PLA         .................
DC3A  PHP         .................
DC3B  LSR A       .................
DC3C  PLP         .................
DC3D  BCC DC41    .................
DC3F  ORA #8Ø     .................
.................  DC61  ASL Ø1,X
.................  DC63  BCC DC67
.................  DC65  INC Ø1,X
.................  DC67  ROR Ø1,X    réajuster octet 1
.................  DC69  ROR Ø1,X    octet 1
.................  DC6B  ROR Ø2,X    octet 3
.................  DC6D  ROR Ø3,X    octet 3
.................  DC6F  ROR Ø4,X    octet 4
.................  DC71  ROR A       et extension
DC41  INY         DC72  INY         plus de décalage ?
DC42  BNE DCØ8    DC73  BNE DC61    si, on repart
DC44  CLC         DC75  CLC         non, sortir, C=Ø
DC45  RTS         DC76  RTS
```

CONSTANTES DIVERSES

```
DC46 DC77  BYT #82,#13,#5D,#8D,#DE  soit  2,3Ø258509  ou encore LN(1Ø)
.... DC7C  BYT #82,#49,#ØF,#DA,#9E  soit  3,14159265  ou encore PI
DC4B DC81  BYT #81,#ØØ,#ØØ,#ØØ,#ØØ  soit  1
```

VALEUR POUR CALCUL DE LN

```
DC5Ø DC86  BYT #Ø3
DC51 DC87  BYT #7F,#7E,#56,#CB,#79  soit  Ø,434255942
DC56 DC8C  BYT #89,#13,#9B,#ØB,#64  soit  Ø,576584541
DC5B DC91  BYT #8Ø,#76,#38,#93,#16  soit  Ø,961800759
DC6Ø DC96  BYT #82,#38,#AA,#3B,#2Ø  soit  2,88539007
```

```
DC65 DC9B  BYT #98,#35,#04,#F3,#34  soit  0,707106781 ou encore SQR(2)/2
DC6A DCA0  BYT #81,#35,#04,#F3,#34  soit  1,41421356  ou encore SQR(2)
DC6F DCA5  BYT #80,#80,#00,#00,#00  soit -0,5
DC74 DCAA  BYT #80,#31,#72,#17,#F8  soit  0,693147181 ou encore LN(2)
```

'LN' (FONCTION)

```
DC79  JSR $DF04   DCAF  JSR $DF13   A=SGN(ACC1)
DC7C  BEQ DC80    DCB2  BEQ DCB6    si 0, erreur
DC7E  BPL DC83    DCB4  BPL DCB9
DC80  JMP $D2A0   DCB6  JMP $D336   si négatif, aussi:'ILLEGAL QUANTITY ERROR'
DC83  LDA D0      DCB9  LDA D0      prendre exposant
DC85  SBC #7F     DCBB  SBC #7F     ramener à 0-127
DC87  PHA         DCBD  PHA         et sauver
DC88  LDA #80     DCBE  LDA #80     exposant=+0
DC8A  STA D0      DCC0  STA D0      et on sauve
DC8C  LDA #65     DCC2  LDA #9B
DC8E  LDY #DC     DCC4  LDY #DC     indexer SQR(2)/2
DC90  JSR $DA97   DCC6  JSR $DB22   (AY)+ACC1 --> ACC1
DC93  LDA #6A     DCC9  LDA #A0
DC95  LDY #DC     DCCB  LDY #DC     indexer SQR(2)
DC97  JSR $DDE0   DCCD  JSR $DDE4   (AY)/ACC1 --> ACC1
DC9A  LDA #4B     DCD0  LDA #81
DC9C  LDY #DC     DCD2  LDY #DC     indexer 1
DC9E  JSR $DA80   DCD4  JSR $DB0B   (AY)-ACC1 --> ACC1
DCA1  LDA #50     DCD7  LDA #86
DCA3  LDY #DC     DCD9  LDY #DC     prendre adresse du polynome
DCA5  JSR $E2F9   DCDB  JSR $E2FD   et calculer
DCA8  LDA #6F     DCDE  LDA #A5
DCAA  LDY #DC     DCE0  LDY #DC     indexer -0,5
DCAC  JSR $DA97   DCE2  JSR $DB22   (AY)+ACC1 --> ACC1
DCAF  PLA         DCE5  PLA         récupérer exposant
DCB0  JSR $E072   DCE6  JSR $E076   A+ACC1 --> ACC1
DCB3  LDA #74     DCE9  LDA #AA
DCB5  LDY #DC     DCEB  LDY #DC     indexer LN(2) et faire multiplication
```

(AY)*ACC1 --> ACC1

```
DCB7  JSR $DD4D   DCED  JSR $DD51   (AY) --> ACC2
```

'*' (OPERATEUR) ACC2*ACC1 --> ACC1

Principe: compte tenu de la représentation des nombres, il suffit d'additionner les exposants, et de faire le produit des mantisses:

Figure DC-C

Le produit de $A=M.2^{E}$ et de $B=M'.2^{E'}$
donne: $A.B=M.M'.2^{E}.2^{E'}$, soit $A.B=M.M'.2^{E+E'}$

La méthode utilisée pour faire le produit des mantisses est assez classique. Deux points méritent d'être soulignés:

-Au lieu de considérer le multiplicateur (ACC1) dans son ensemble, il est pris octet par octet, ce qui réduit sensiblement le nombre de décalages, qui est de 5 fois 8 décalages sur un octet, soit 40, contre 40 décalages sur 5 octets, soit 200 décalages (avec cette dernière méthode, on décale pour rien des octets nuls).

-D'habitude on décale le multiplicande à gauche, et on additionne ou non au résultat (Figure DC-A). Cette fois, c'est le résultat qui est décalé à droite, les décalages du multiplicande se faisant alors vers la droite. Le résultat est le même, mais on n'utilise une zone temporaire de la taille du résultat, au lieu du double avec la première méthode.

-Il s'ensuit tout de même une perte de précision, car les bits ainsi perdus auraient pu, en s'additionnant, influencer sur la mantisse. C'est pour ceci que des opérations entre entiers donnent parfois des résultats approximatifs ! (Figure DC-B)

Figure DC-A

Exemple: Acc1 et Acc2 sont réduits à 4 bits, résultat sur 4 bits:
b3 b2 b1 b0 a3 a2 a1 a0

```
Multiplicande (Acc2)      a3 a2 a1 a0
Multiplicateur (Acc1)     b3 b2 b1 b0
                          ---------------
                          a3 a2 a1 a0      addition si b0=1
                       a3 a2 a1 a0         addition si b1=1
                    a3 a2 a1 a0            addition si b2=1
                 a3 a2 a1 a0               addition si b3=1
                 ---------------
Résultat:        r3 r2 r1 r0
```

Le résultat est sur 4 bits, mais le calcul a nécessité 7 bits.

Figure DC-B

Exemple: Acc1 et Acc2 sont réduits à 4 bits, résultat sur 4 bits:
b3 b2 b1 b0 a3 a2 a1 a0

```
Multiplicande (Acc2)      a3 a2 a1 a0
Multiplicateur (Acc1)     b3 b2 b1 b0
                          ---------------
                          a3 a2 a1 a0      addition si b0=1
                       a3 a2 a1 a0         addition si b1=1
                    a3 a2 a1 a0            addition si b2=1
                 a3 a2 a1 a0               addition si b3=1
                 ---------------
Résultat:        r3 r2 r1 r0
```

Le résultat est sur 4 bits, mais le calcul a nécessité 4 bits aussi .

```
DCBA  BNE DCBF    DCFØ  BNE DCF5
DCBC  JMP $DD4C   DCF2  JMP $DD5Ø   si ACC1=Ø, c'est le résultat, et on sort
DCBF  JSR $DD78   DCF5  JSR $DD7C   faire la somme des exposants
```

Faire le produit des mantisses

```
DCC2  LDA #ØØ     DCF8  LDA #ØØ
DCC4  STA 95      DCFA  STA 95
DCC6  STA 96      DCFC  STA 96      vider zone résultat
DCC8  STA 97      DCFE  STA 97      (notée ACC3)
DCCA  STA 98      DDØØ  STA 98
DCCC  LDA DF      DDØ2  LDA DF      prendre extension
DCCE  JSR $DCE8   DDØ4  JSR $DD1E   et multiplier par ACC2
DCD1  LDA D4      DDØ7  LDA D4      octet 4
DCD3  JSR $DCE8   DDØ9  JSR $DD1E   multiplier aussi
DCD6  LDA D3      DDØC  LDA D3      octet 3
DCD8  JSR $DCE8   DDØE  JSR $DD1E   multiplier
DCDB  LDA D2      DD11  LDA D2      octet 2
DCDD  JSR $DCE8   DD13  JSR $DD1E   multiplier
DCEØ  LDA D1      DD16  LDA D1      octet 1
DCE2  JSR $DCED   DD18  JSR $DD23   multiplier
DCE5  JMP $DE6Ø   DD1B  JMP $DE64   ACC3 --) ACC1 et justification
```

ACC3+ACC2*A --) ACC3

Programmation: une astuce est utilisée pour éviter un compteur qui décompterait les huit décalages: le multiplicande voit son b7 forcé à 1, de sorte qu'il ne sera nul qu'après huit décalages à droite.

```
DCE8  BNE DCED    DD1E  BNE DD23   si A=Ø, on décale d'un octet vers la droite
DCEA  JMP $DBE5   DD2Ø  JMP $DC3E  i.e. faire huit décalages
DCED  LSR A       DD23  LSR A      sortir premier bit (bØ)
DCEE  ORA #8Ø     DD24  ORA #8Ø    et forcer b7 à 1 (compter 8 décalages)
DCFØ  TAY         DD26  TAY        sauver dans Y multiplicande
DCF1  BCC DDØC    DD27  BCC DD42   si Ø sorti, décaler simplement
```

Additionner résultat provisoire et multiplicande

```
DCF3  CLC         DD29  CLC
DCF4  LDA 98      DD2A  LDA 98      ajouter octet 4
DCF6  ADC DC      DD2C  ADC DC
DCF8  STA 98      DD2E  STA 98
```

```
DCFA  LDA 97      DD3Ø  LDA 97
DCFC  ADC DB      DD32  ADC DB      octet 3
DCFE  STA 97      DD34  STA 97
DDØØ  LDA 96      DD36  LDA 96
DDØ2  ADC DA      DD38  ADC DA      octet 2
DDØ4  STA 96      DD3A  STA 96
DDØ6  LDA 95      DD3C  LDA 95
DDØ8  ADC D9      DD3E  ADC D9      octet 1
DDØA  STA 95      DD4Ø  STA 95
```

Décaler le résultat provisoire

```
DDØC  LDA #ØØ     ..............
DDØE  BCC DD12    ..............
DD1Ø  LDA #8Ø     ..............
DD12  LSR 95      ..............
DD14  ORA 95      ..............
DD16  STA 95      ..............
DD18  LDA #ØØ     ..............
DD1A  BCC DD1E    ..............
DD1C  LDA #8Ø     ..............
DD1E  LSR 96      ..............
DD2Ø  ORA 96      ..............
DD22  STA 96      ..............
DD24  LDA #ØØ     ..............  pourquoi faire simple ?
DD26  BCC DD2A    ..............
DD28  LDA #8Ø     ..............
DD2A  LSR 97      ..............
DD2C  ORA 97      ..............
DD2E  STA 97      ..............
DD3Ø  LDA #ØØ     ..............
DD32  BCC DD36    ..............
DD34  LDA #8Ø     ..............
DD36  LSR 98      ..............
DD38  ORA 98      ..............
DD3A  STA 98      ..............
DD3C  LDA #ØØ     ..............
DD3E  BCC DD42    ..............
DD4Ø  LDA #8Ø     ..............
DD42  LSR DF      ..............
DD44  ORA DF      ..............
DD46  STA DF      ..............
..............    DD42  ROR 95      octet 1
..............    DD44  ROR 96      octet 2
```

```
.....  DD46  ROR 97      octet 3
.....  DD48  ROR 98      octet 4
.....  DD4A  ROR DF      octet extension
DD48  TYA     DD4C  TYA      récupérer multiplicande
DD49  LSR A   DD4D  LSR A    et sortir le bit
DD4A  BNE DCF0  DD4E  BNE DD26  et continuer si pas 8 décalages
DD4C  RTS     DD50  RTS
```

(AY) --> ACC2

Entrée: AY contient l'adresse de la valeur (signe incorporé à octet 1)

```
DD4D  STA 91      DD51  STA 91
DD4F  STY 92      DD53  STY 92      sauver adresse de la valeur
DD51  LDY #04     DD55  LDY #04     indexer octet 4
DD53  LDA (91),Y  DD57  LDA (91),Y
DD55  STA DC      DD59  STA DC      octet 4
DD57  DEY         DD5B  DEY
DD58  LDA (91),Y  DD5C  LDA (91),Y
DD5A  STA DB      DD5E  STA DB      octet 3
DD5C  DEY         DD60  DEY
DD5D  LDA (91),Y  DD61  LDA (91),Y
DD5F  STA DA      DD63  STA DA      octet 2
DD61  DEY         DD65  DEY
DD62  LDA (91),Y  DD66  LDA (91),Y
DD64  STA DD      DD68  STA DD      octet 1
DD66  EOR D5      DD6A  EOR D5
DD68  STA DE      DD6C  STA DE      et ajuster produit des signes
DD6A  LDA DD      DD6E  LDA DD
DD6C  ORA #80     DD70  ORA #80     forcer b7 octet 1
DD6E  STA D9      DD72  STA D9
DD70  DEY         DD74  DEY
DD71  LDA (91),Y  DD75  LDA (91),Y
DD73  STA D8      DD77  STA D8      exposant
DD75  LDA D0      DD79  LDA D0      et positionner Z selon ACC1
DD77  RTS         DD7B  RTS
```

FAIRE SOMME DES EXPOSANTS

Remarque: cette routine est appelée aussi bien par la multiplication que par la division. Dans les deux cas, le résultat est nul si ACC2. Ce cas est traité à part, pour éviter des erreurs d'arrondi... qui seraient désastreuses pour l'opinion.

Principe: les joies des additions en arithmétiques signée, et inversée de plus.

Rappel:#81=Ø, #82=1, #8Ø=-1. Voir le chapitre sur le codage des nombres our des précisions sur la notation de l'exposant.

Le tableau suivant récapitule avec des exemples typiques tous les cas possibles.

Figure DC-D

Exposant 1	Exposant 2	Somme	Vrai résultat	Signe	Dépassement	Overflow
#90 (+15)	#90 (+15)	#20	#A0 (+31)	0	1	0
#C1 (+64)	#C0 (+63)	#81	#101 (128)	1	1	1
#50 (-49)	#80 (-1)	#D0	#50 (-49)	1	0	0
#20 (-97)	#20 (-97)	#40	#C0 (-191)	0	0	0

```
DD78  LDA D8      DD7C  LDA D8      prendre exposant ACC2
DD7A  BEQ DD9B    DD7E  BEQ DD9F    si nul, faire ACC1=Ø
DD7C  CLC         DD8Ø  CLC
DD7D  ADC DØ      DD81  ADC DØ      ajouter exposant ACC1
DD7F  BCC DD85    DD83  BCC DD89    si pas de dépassement, sauter
DD81  BMI DDAØ    DD85  BMI DDA4    dépassement et négatif:Overflow
DD83  CLC         DD87  CLC         ajuster C
DD84  BYT #2C     DD88  BYT #2C
DD85  BPL DD9B    DD89  BPL DD9F    pas de dépassement et positif: Underflow
DD87  ADC #8Ø     DD8B  ADC #8Ø
DD89  STA DØ      DD8D  STA DØ      récupérer le bon résultat
DD8B  BNE DD9Ø    DD8F  BNE DD94    si résultat nul,
DD8D  JMP $DB2B   DD91  JMP $DBB6   indiquer signe positif
DD9Ø  LDA DE      DD94  LDA DE      Sinon, le signe du résultat
DD92  STA D5      DD96  STA D5      est celui du produit des signes
DD94  RTS         DD98  RTS

DD95  LDA D5      DD99  LDA D5
DD97  EOR #FF     DD9B  EOR #FF
DD99  BMI DDAØ    DD9D  BMI DDA4
DD9B  PLA         DD9F  PLA         enlever adresse de retour
DD9C  PLA         DDAØ  PLA         pour retourner directement à appel de ↓
DD9D  JMP $DB27   DDA1  JMP $DBB2   et ACC1=Ø
DDAØ  JMP $DBEØ   DDA4  JMP $DC39   'OVERFLOW ERROR'
```

10↓ACC1 --> ACC1

Principe: au lieu d'utiliser la routine de multiplication, un simple travail

sur l'exposant et une seule addition permettent d'arriver au même résultat, plus rapidement. C'est important,puisque c'est la routine clé de la conversion décimal --) flottant.

```
DDA3  JSR $DEDD   DDA7  JSR $DEE5   AACC1 --) ACC2
DDA6  TAX         DDAA  TAX         positionner Z selon exposant
DDA7  BEQ DDB9    DDAB  BEQ DDBD    si nombre nul, résultat aussi
DDA9  CLC         DDAD  CLC
DDAA  ADC #02     DDAE  ADC #02     *4 :exposant+2
DDAC  BCS DDA0    DDB0  BCS DDA4   sortir si overflow
DDAE  LDX #00     DDB2  LDX #00     produit des signes=0
DDB0  STX DE      DDB4  STX DE
DDB2  JSR $DAA7   DDB6  JSR $DB32   *5 :ACC2+ACC1 --) ACC1
DDB5  INC D0      DDB9  INC D0      *10:exposant+1 (soit multiplier par 2)
DDB7  BEQ DDA0    DDBB  BEQ DDA4    sortir si overflow
DDB9  RTS         DDBD  RTS

DDBA DDBE  BYT #84,#20,#00,#00,#00  soit  10
```

ACC1/10 --) ACC1

```
DDBF  JSR $DEDD   DDC3  JSR $DEE5   AACC1 --) ACC2
DDC2  LDA #BA     DDC6  LDA #BE
DDC4  LDY #DD     DDC8  LDY #DD     indexer valeur 10
DDC6  LDX #00     DDCA  LDX #00
DDC8  STX DE      DDCC  STX DE      indiquer produit des signes=0
DDCA  JSR $DE73   DDCE  JSR $DE7B   (AY) --) ACC1
DDCD  JMP $DDE3   DDD1  JMP $DDE7   et faire ACC2/ACC1
```

'LOG' (FONCTION)

Principe: la classique formule (FIGURE DD-A) est utilisée.

$$\log_{10}(x) = \frac{\ln(x)}{\ln(10)}$$

Figure DD-A

```
DDD0  JSR $DC79   DDD4  JSR $DCAF   faire LN
DDD3  JSR $DEDD   DDD7  JSR $DEE5   AACC1 --) ACC2
DDD6  LDA #46     DDDA  LDA #77
DDD8  LDY #DC     DDDC  LDY #DC     indexer LN(10)
DDDA  JSR $DE73   DDDE  JSR $DE7B   (AY) --) ACC1
DDDD  JMP $DDE3   DDE1  JMP $DDE7   et faire ACC2/ACC1
```

(AY)/ACC1 --) ACC1

```
DDEØ  JSR $DD4D   DDE4  JSR $DD51
```

'/' (OPERATEUR) ACC2/ACC1 --> ACC1

Principe: Comme pour la multiplication, le calcul est en fait ramené à la division des mantisses, et à leur justification: (Figure DD-B).

Pour soustraire les exposants, on calcule en fait l'opposé de l'exposant de ACC1: ajouter l'opposé, c'est soustraire.

La division des mantisses est formellement compliquée. Son principe est simple, c'est le même qu'en décimal. C'est même plus simple, puisqu'en décimal, il faut essayer les chiffres de Ø à 9, alors qu'en binaire, il n'y a que deux possibilités (Ø ou 1). On remarque alors que le résultat est celui de la différence du multiplicande et du multiplicateur.

Programmation: la routine manque de 'finition', de nombreux branchements pourraient être évités. Cela n'a pas d'incidence sur le temps d'exécution, mais on aurait pu gagner une bonne dizaine d'octet.

```
DDE3  BEQ DE5B    DDE7  BEQ DE5F    si ACC1=Ø,'DIVISION BY ZERO ERROR'
DDE5  JSR $DEEC   DDE9  JSR $DEF4   AACC1 --> ACC1
DDE8  LDA #ØØ     DDEC  LDA #ØØ
DDEA  SEC         DDEE  SEC
DDEB  SBC DØ      DDEF  SBC DØ      complémenter exposant
DDED  STA DØ      DDF1  STA DØ
DDEF  JSR $DD78   DDF3  JSR $DD7C   puis faire somme des exposants
DDF2  INC DØ      DDF6  INC DØ      ajuster
DDF4  BEQ DDAØ    DDF8  BEQ DDA4    et Overflow si tombe à Ø
DDF6  LDX #FC     DDFA  LDX #FC     indexer ACC3 (#98+#FC+1=#195,soit #95)
DDF8  LDA #Ø1     DDFC  LDA #Ø1     initialiser pour 8 décalages
DDFA  LDY D9      DDFE  LDY D9
DDFC  CPY D1      DEØØ  CPY D1      comparer ACC1 et ACC2
DDFE  BNE DE1Ø    DEØ2  BNE DE14    sauter si différent
DEØØ  LDY DA      DEØ4  LDY DA
DEØ2  CPY D2      DEØ6  CPY D2
DEØ4  BNE DE1Ø    DEØ8  BNE DE14    idem octet 2
DEØ6  LDY DB      DEØA  LDY DB
DEØ8  CPY D3      DEØC  CPY D3
DEØA  BNE DE1Ø    DEØE  BNE DE14    idem octet 3
DEØC  LDY DC      DE1Ø  LDY DC
DEØE  CPY D4      DE12  CPY D4      idem octet 4

DE1Ø  PHP         DE14  PHP         sauver signe comparaison (dans C)
DE11  ROL A       DE15  ROL A       assembler le résultat
```

La division de $A = M.2^E$ par $B = M'.2^{E'}$

donne : $\frac{M.2^E}{M'.2^{E'}} = (M / M').2^{E-E'}$

Figure DD-B

```
DE12  BCC DE1D    DE16  BCC DE21    sauter si pas encore 8 décalages
DE14  INX         DE18  INX         ajuster index, et placer N et Z
DE15  STA 98,X    DE19  STA 98,X    sauver octet résulat
DE17  BEQ DE4B    DE1B  BEQ DE4F    index=0, on passe à l'extension
DE19  BPL DE4F    DE1D  BPL DE53    index positif, c'est fini
DE1B  LDA #01     DE1F  LDA #01     indiquer à nouveau 8 décalages
DE1D  PLP         DE21  PLP         récupérer sens comparaison
DE1E  BCS DE2E    DE22  BCS DE32    Dividende > diviseur: soustraction
```

Décaler dividende (ACC2), soit diviser par 2.

```
DE20  ASL DC      DE24  ASL DC      octet 4
DE22  ROL DB      DE26  ROL DB      octet 3
DE24  ROL DA      DE28  ROL DA      octet 2
DE26  ROL D9      DE2A  ROL D9      octet 1
DE28  BCS DE10    DE2C  BCS DE14    1 sort: ACC2 obligatoirement > ACC1
DE2A  BMI DDFA    DE2E  BMI DDFE    faire comparaison, on n'est pas sur
DE2C  BPL DE10    DE30  BPL DE14
```

Calculer dividende-diviseur

```
DE2E  TAY         DE32  TAY         sauver résultat provisoire
DE2F  LDA DC      DE33  LDA DC
DE31  SBC D4      DE35  SBC D4
DE33  STA DC      DE37  STA DC      soustraction octet 4
DE35  LDA DB      DE39  LDA DB
DE37  SBC D3      DE3B  SBC D3
DE39  STA DB      DE3D  STA DB      soustraire octet 3
DE3B  LDA DA      DE3F  LDA DA
DE3D  SBC D2      DE41  SBC D2
DE3F  STA DA      DE43  STA DA      soustraire octet 2
DE41  LDA D9      DE45  LDA D9
DE43  SBC D1      DE47  SBC D1
DE45  STA D9      DE49  STA D9      et enfin octet 1
DE47  TYA         DE4B  TYA         récupérer résultat
DE48  JMP $DE20   DE4C  JMP $DE24   ou... BCS:décaler dividende
```

Calculer pour extension

```
DE4B  LDA #40     DE4F  LDA #40     ne faire que 2 décalages
DE4D  BNE DE1D    DE51  BNE DE21    inconditionnel: continuer
```

Finir une division

```
DE4F  ASL A       DE53  ASL A
DE50  ASL A       DE54  ASL A
DE51  ASL A       DE55  ASL A
DE52  ASL A       DE56  ASL A
DE53  ASL A       DE57  ASL A       ramener résultat division extension
DE54  ASL A       DE58  ASL A       dans b7 b6
DE55  STA DF      DE59  STA DF      et sauver extension
DE57  PLP         DE5B  PLP         ajuster la pile
DE58  JMP $DE60   DE5C  JMP $DE64   ACC3 --> ACC1

DE5B  LDX #85     DE5F  LDX #85
DE5D  JMP $C485   DE61  JMP $C47E   'DIVISION BY ZERO'
```

ACC3 --> ACC1

```
DE60  LDA 95      DE64  LDA 95      transfert octet 1
DE62  STA D1      DE66  STA D1
DE64  LDA 96      DE68  LDA 96      octet 2
DE66  STA D2      DE6A  STA D2
DE68  LDA 97      DE6C  LDA 97      octet 3
DE6A  STA D3      DE6E  STA D3
DE6C  LDA 98      DE70  LDA 98      octet 4
DE6E  STA D4      DE72  STA D4
DE70  JMP $DB07   DE74  JMP $DB92   et justifier la mantisse
```

'PI' (FONCTION)

Programmation: la routine placée là évite un JMP, c'est un bon exemple d'optimisation, classique il est vrai.

```
.............  DE77  LDA #7C
.............  DE79  LDY #DC      indexer la valeur PI
```

(AY) --> ACC1

```
DE73  STA 91      DE7B  STA 91
DE75  STY 92      DE7D  STY 92      sauver adresse
DE77  LDY #04     DE7F  LDY #04     indexer octet 4
DE79  LDA (91),Y  DE81  LDA (91),Y
```

```
DE7B  STA D4      DE83  STA D4      octet 4
DE7D  DEY         DE85  DEY
DE7E  LDA (91),Y  DE86  LDA (91),Y
DE80  STA D3      DE88  STA D3      octet 3
DE82  DEY         DE8A  DEY
DE83  LDA (91),Y  DE8B  LDA (91),Y
DE85  STA D2      DE8D  STA D2      octet 2
DE87  DEY         DE8F  DEY
DE88  LDA (91),Y  DE90  LDA (91),Y
DE8A  STA D5      DE92  STA D5      octet 1 (sauver le signe)
DE8C  ORA #80     DE94  ORA #80     forcer b7
DE8E  STA D1      DE96  STA D1      dans octet 1
DE90  DEY         DE98  DEY
DE91  LDA (91),Y  DE99  LDA (91),Y
DE93  STA D0      DE9B  STA D0      exposant
DE95  STY DF      DE9D  STY DF      extension=0
DE97  RTS         DE9F  RTS


                              AACC1 --) #CB-#CF (ACC5)

DE98  LDX #CB     DEA0  LDX #CB
DE9A  BYT #2C     DEA2  BYT #2C

                              AACC1 --) #C6-#CA (ACC4)

DE9B  LDX #C6     DEA3  LDX #C6
DE9D  LDY #00     DEA5  LDY #00
DE9F  BEQ DEA5    DEA7  BEQ DEAD    inconditionnel


                                 AACC1 --) (#00B8)

DEA1  LDX B8      DEA9  LDX B8      prendre adresse
DEA3  LDY B9      DEAB  LDY B9      dans XY


                                 AACC1 --) XY-XY+4

Sortie:Y=0

DEA5  JSR $DEEC   DEAD  JSR $DEF4   arrondir ACC1 (XY inchangés)

                                 ACC1 --) XY-XY+4
```

```
DEA8  STX 91      DEBØ  STX 91
DEAA  STY 92      DEB2  STY 92      sauver adresse
DEAC  LDY #Ø4     DEB4  LDY #Ø4     indexer octet 4
DEAE  LDA D4      DEB6  LDA D4
DEBØ  STA (91),Y  DEB8  STA (91),Y  octet 4
DEB2  DEY         DEBA  DEY
DEB3  LDA D3      DEBB  LDA D3
DEB5  STA (91),Y  DEBD  STA (91),Y  octet 3
DEB7  DEY         DEBF  DEY
DEB8  LDA D2      DECØ  LDA D2
DEBA  STA (91),Y  DEC2  STA (91),Y  octet 2
DEBC  DEY         DEC4  DEY
DEBD  LDA D5      DEC5  LDA D5      prendre signe
DEBF  ORA #7F     DEC7  ORA #7F     forcer bits non significatifs
DEC1  AND D1      DEC9  AND D1      et positionner b7 (signe) et le reste
DEC3  STA (91),Y  DECB  STA (91),Y  selon octet 1
DEC5  DEY         DECD  DEY
DEC6  LDA DØ      DECE  LDA DØ
DEC8  STA (91),Y  DEDØ  STA (91),Y  exposant
DECA  STY DF      DED2  STY DF      extension=Ø
DECC  RTS         DED4  RTS
```

ACC2 --> ACC1

Remarque:Ne touche pas à Y

```
DECD  LDA DD      DED5  LDA DD      signe ACC2
DECF  STA D5      DED7  STA D5      dans signe ACC1
DED1  LDX #Ø5     DED9  LDX #Ø5
DED3  LDA D7,X    DEDB  LDA D7,X    prendre ACC2
DED5  STA CF,X    DEDD  STA CF,X    dans ACC1
DED7  DEX         DEDF  DEX
DED8  BNE DED3    DEEØ  BNE DEDB
DEDA  STX DF      DEE2  STX DF      extension=Ø
DEDC  RTS         DEE4  RTS
```

AACC1 --> ACC2

```
DEDD  JSR $DEEC   DEE5  JSR $DEF4   arrondir ACC1
```

ACC1 --> ACC2

```
DEEØ  LDX #Ø6     DEE8  LDX #Ø6     décaler 6 octets
DEE2  LDA CF,X    DEEA  LDA CF,X    prendre ACC1
DEE4  STA D7,X    DEEC  STA D7,X    dans ACC2
DEE6  DEX         DEEE  DEX
DEE7  BNE DEE2    DEEF  BNE DEEA    jusqu'à l'exposant...
DEE9  STX DF      DEF1  STX DF      extension=Ø
DEEB  RTS         DEF3  RTS
```

AACC1 --) ACC1 (SELON EXTENSION)

Principe: selon b7 de l'extension, la mantisse est incrémentée ou non. C'est comme, en décimal, lorsqu'on veut arrondir un nombre décimal: on ajoute Ø,5 et on prend la partie entière.

```
DEEC  LDA DØ      DEF4  LDA DØ      si nul,résultat aussi
DEEE  BEQ DEEB    DEF6  BEQ DEF3
DEFØ  ASL DF      DEF8  ASL DF      prendre bit d'extension
DEF2  BCC DEEB    DEFA  BCC DEF3    si Ø,rien á faire: on sort
DEF4  JSR $DBD1   DEFC  JSR $DC2A   sinon incrémenter la mantisse
DEF7  BNE DEEB    DEFF  BNE DEF3    si pas de report, on sort
DEF9  JMP $DB68   DFØ1  JMP $DBF3   sinon, incrémenter exposant
```

YA --) ACC1 (NON SIGNE)

```
.............  DFØ4  JSR $D2A9   YA --) ACC1 (non signé)
.............  DFØ7  LSR D4
.............  DFØ9  BCS DFØF
```

'FALSE' (FONCTION)

```
DEFC  LDA #ØØ     DFØB  LDA #ØØ     indiquer résultat=Ø
DEFE  BEQ DF15    DFØD  BEQ DF24    A --) ACC1 (signé)
```

'TRUE' (FONCTION)

```
DFØØ  LDA #FF     DFØF  LDA #FF     indiquer résultat=-1
DFØ2  BMI DF15    DF11  BMI DF24    A --) ACC1 (signé)
```

A=SGN (ACC1)

Entrée: le nombre est dans ACC1 (signe dans #D5)

```
Sortie: A=#ØØ,Z=1,C=?,N=Ø si ACC1=Ø
        A=#FF,Z=Ø,C=1,N=1 si ACC1<Ø
        A=#Ø1,Z=Ø,C=Ø,N=Ø si ACC1>Ø

DFØ4  LDA DØ      DF13  LDA DØ      prendre exposant
DFØ6  BEQ DF11    DF15  BEQ DF2Ø    si nul,nombre=Ø et signe=Ø
DFØ8  LDA D5      DF17  LDA D5      prendre signe
DFØA  ROL A       DF19  ROL A       et le signe dans C maintenant
DFØB  LDA #FF     DF1A  LDA #FF     préparer pour signe négatif
DFØD  BCS DF11    DF1C  BCS DF2Ø    et on sort si oui
DFØF  LDA #Ø1     DF1E  LDA #Ø1     indiquer signe plus
DF11  RTS         DF2Ø  RTS
```

SGN (FONCTION)

```
DF12  JSR $DFØ4   DF21  JSR $DF13   A=signe ACC1
```

A --) ACC1 (SIGNE)

```
DF15  STA D1      DF24  STA D1      on sauve le nombre
DF17  LDA #ØØ     DF26  LDA #ØØ     annuler le poids faible
DF19  STA D2      DF28  STA D2
DF1B  LDX #88     DF2A  LDX #88     exposant base 2=7
DF1D  LDA D1      DF2C  LDA D1      et on prend le signe
DF1F  EOR #FF     DF2E  EOR #FF     on inverse le signe
DF21  ROL A       DF3Ø  ROL A       dans C
```

#D2-#D1 --) ACC1

Entrée: #D2-#D1 contient le nombre, et C=Ø si négatif, C=1 si positif, X contient l'exposant désiré

Sortie: nombre en flottant dans ACC1

Principe: si C=Ø,le nombre va être complémenté, c'est à dire transformé en nombre négatif

```
DF22  LDA #ØØ     DF31  LDA #ØØ     et on annule le reste de la mantisse
DF24  STA D4      DF33  STA D4      octet 4
DF26  STA D3      DF35  STA D3      octet 3
DF28  STX DØ      DF37  STX DØ      placer exposant
```

```
DF2A  STA DF      DF39  STA DF      produit des signes nul
DF2C  STA D5      DF3B  STA D5      signe nul pour l'instant aussi
DF2E  JMP $DB02   DF3D  JMP $DB8D   ajuster la mantisse selon l'exposant
```

YA --) ACC1 (NON SIGNE)

```
.............     DF40  STA D1      sauver poids fort
.............     DF42  STY D2      puis poids faible
.............     DF44  LDX #90     exposant désiré =+15
.............     DF46  SEC         indiquer positif
.............     DF47  BCS DF31    et finir
```

'ABS' (FONCTION)

Remarque: la fonction la plus courte...

```
DF31  LSR D5      DF49  LSR D5      indiquer ACC1 positif
DF33  RTS         DF4B  RTS         et c'est tout !
```

COMPARER ACC1 ET (AY) (SOIT SGN (ACC1-(AY))

Sortie: voir 'A=SGN(ACC1)'

Principe: simple, mais formellement compliqué par le fait que ACC1 est dans sa forme étendue (signe dans #D5), alors que l'autre opérande ne l'est pas.

```
DF34  STA 93      DF4C  STA 93      sauver adresse de l'opérande
DF36  STY 94      DF4E  STY 94
DF38  LDY #00     DF50  LDY #00
DF3A  LDA (93),Y  DF52  LDA (93),Y  prendre exposant
DF3C  INY         DF54  INY         indexer la mantisse
DF3D  TAX         DF55  TAX         dans X
DF3E  BEQ DF04    DF56  BEQ DF13    si nul,résultat pareil que ACC1
DF40  LDA (93),Y  DF58  LDA (93),Y  prendre le signe
DF42  EOR D5      DF5A  EOR D5      est-ce la même que celui de ACC1 ?
DF44  BMI DF08    DF5C  BMI DF17    non,résultat pareil que ACC1
DF46  CPX D0      DF5E  CPX D0      oui,comparer à exposant de ACC1
DF48  BNE DF6B    DF60  BNE DF83    et résultat dans C si pas pareil
DF4A  LDA (93),Y  DF62  LDA (93),Y  reprendre mantisse
DF4C  ORA #80     DF64  ORA #80     et forcer signe (non significatif
DF4E  CMP D1      DF66  CMP D1      car pour ACC1 le signe est dans #D5)
```

```
DF5Ø  BNE DF6B    DF68  BNE DF83    résultat dans C
DF52  INY         DF6A  INY         octet 2 mantisse
DF53  LDA (93),Y  DF6B  LDA (93),Y
DF55  CMP D2      DF6D  CMP D2
DF57  BNE DF6B    DF6F  BNE DF83
DF59  INY         DF71  INY
DF5A  LDA (93),Y  DF72  LDA (93),Y  octet 3 mantisse
DF5C  CMP D3      DF74  CMP D3
DF5E  BNE DF6B    DF76  BNE DF83
DF6Ø  INY         DF78  INY
DF61  LDA #7F     DF79  LDA #7F     placer signe dans C
DF63  CMP DF      DF7B  CMP DF      selon le bit d'extension
DF65  LDA (93),Y  DF7D  LDA (93),Y  octet 4
DF67  SBC D4      DF7F  SBC D4      sortir Z=1,C=1,A=Ø si résultat nul
DF69  BEQ DF93    DF81  BEQ DFAB
DF6B  LDA D5      DF83  LDA D5      prendre signe ACC1
DF6D  BCC DF71    DF85  BCC DF89    si ACC1 > (AY),résultat pareil que ACC1
DF6F  EOR #FF     DF87  EOR #FF     sinon signe inverse
DF71  JMP $DFØA   DF89  JMP $DF19   et prendre le signe (ou inverse) de ACC1
```

ACC1 --> #D4-#D3-#D2-#D1 (SIGNE)

Sortie: #D7=Y=Ø

Principe: on commence par vérifier que c'est possible, grâce à l'exposant. De même, le résultat étant signé, on complémente éventuellement.

Il suffit après de décaler la mantisse afin d'obtenir l'exposant désiré (31). A cet effet, il faudra entrer des Ø à gauche pour un nombre positif, et des 1 pour un nombre négatif. #D7 sert à le préciser.

#D7 est supposé nul, puiqu'il n'est initialisé que s'il doit être égal à #FF. Donc, si il n'est pas nul, le décalage sera erroné: POKE #D7,1: PRINT 1 donne...1,ØØØ65793 !

```
DF74  LDA D0      DF8C  LDA D0      prendre exposant
DF76  BEQ DFC2    DF8E  BEQ DFDA    si nul, mantisse nulle aussi
DF78  SEC         DF9Ø  SEC
DF79  SBC #AØ     DF91  SBC #AØ     enlever exposant=+31
DF7B  BIT D5      DF93  BIT D5      tester signe de ACC1
DF7D  BPL DF88    DF95  BPL DFAØ    si positif,c'est bon
DF7F  TAX         DF97  TAX         sinon,excédent exposant dans X
DF8Ø  LDA #FF     DF98  LDA #FF     préparer pour remplir avec des 1
DF82  STA D7      DF9A  STA D7
DF84  JSR $DBAF   DF9C  JSR $DCØ8   et complémenter la mantisse
```

```
DF87  TXA          DF9F  TXA          récupérer excédent dans A
DF88  LDX #D0      DFA0  LDX #D0      préparer index pour rotations
DF8A  CMP #F9      DFA2  CMP #F9      y-a-t-il plus de 8 décalage ?
DF8C  BPL DF94     DFA4  BPL DFAC     non,on saute
DF8E  JSR $DBFB    DFA6  JSR $DC54    oui,on fait les décalages
DF91  STY D7       DFA9  STY D7       et on récupère la valeur de justification
DF93  RTS          DFAB  RTS
```

Faire 8 décalages de la mantisse

```
DF94  TAY          DFAC  TAY          sauver excédent exposant
DF95  LDA D5       DFAD  LDA D5       prendre signe
DF97  AND #80      DFAF  AND #80      dans b7
DF99  LSR D1       DFB1  LSR D1       on fait premier décalage
DF9B  ORA D1       DFB3  ORA D1       et on inclut le signe
DF9D  STA D1       DFB5  STA D1       et on sauve
DF9F  JSR $DC14    DFB7  JSR $DC6B    on décale
DFA2  STY D7       DFBA  STY D7       Y=0
DFA4  RTS          DFBC  RTS
```

'INT' (FONCTION)

Principe: on converti ACC1 en entier, si c'est possible (si ACC1 est trop grand, c'est déjà un entier). Convertir en entier revient à se ramener à un exposant nul.

```
DFA5  LDA D0       DFBD  LDA D0       prendre exposant
DFA7  CMP #A0      DFBF  CMP #A0      si supérieur à 31,rien à faire
DFA9  BCS DFCB     DFC1  BCS DFE3
DFAB  JSR $DF74    DFC3  JSR $DF8C    convertir en entier en #D4-#D3-#D2-#D1
DFAE  STY DF       DFC6  STY DF       nouvelle extension=0
DFB0  LDA D5       DFC8  LDA D5       prendre signe
DFB2  STY D5       DFCA  STY D5       indiquer signe positif pour l'instant
DFB4  EOR #80      DFCC  EOR #80      inverser le signe
DFB6  ROL A        DFCE  ROL A        dans C (préparer la justification)
DFB7  LDA #A0      DFCF  LDA #A0      indiquer exposant=31
DFB9  STA D0       DFD1  STA D0
DFBB  LDA D4       DFD3  LDA D4       prendre poids faible
DFBD  STA 24       DFD5  STA 24       dans #24 (b0=parité du nombre, pour '^')
DFBF  JMP $DB02    DFD7  JMP $DB8D    justifer la mantisse selon l'exposant et C
```

Annuler la mantisse

```
DFC2  STA D1      DFDA  STA D1      octet 1
DFC4  STA D2      DFDC  STA D2      octet 2
DFC6  STA D3      DFDE  STA D3      octet 3
DFC8  STA D4      DFE0  STA D4      octet 4
DFCA  TAY         DFE2  TAY         et justification=0
DFCB  RTS         DFE3  RTS
DFCC  JMP $E848   DFE4  JMP $E981   aller prendre un nombre Hexa dans ACC1
```

RAMASSER UN NOMBRE DANS ACC1

Entrée: A contient le premier caractère de l'expression, C=0 si chiffre, 1 sinon. Cette routine est aussi appelée par la fonction VAL, ce qui explique que que les signes - ou + ne soient pas obligatoirement codés.

Sortie: ACC1 contient le nombre, #E9 pointe sur le premier caractère qui n'a pu être interprété.

Variables systèmes utilisées:

- #CC: nombre de chiffre après la virgule
- #CD: exposant (base 10)
- #CE: b7=1 si on est dans la partie décimale du nombre
- #CF: b7=signe de l'exposant
- #D0-#D5: accumulateur flottant
- #D6: signe de l'accu (0 ou #FF)

```
EFCF  LDY #00     DFE7  LDY #00
DFD1  LDX #0A     DFE9  LDX #0A     vider la zone #CC-#D6
DFD3  STY CC,X    DFEB  STY CC,X    c'est à dire ACC 1 et divers pointeur
DFD5  DEX         DFED  DEX
DFD6  BPL DFD3    DFEE  BPL DFEB    sortir :X=#FF
DFD8  BCC DFED    DFF0  BCC E005    sauter si chiffre
DFDA  CMP #'#'    DFF2  CMP #'#'    est-ce l'indicateur d'un nombre Hexa ?
DFDC  BEQ DFCC    DFF4  BEQ DFE4    oui, évaluer nombre Hexa
DFDE  CMP #'-'    DFF6  CMP #'-'    est-ce l'incateur d'un nombre négatif ?
DFE0  BNE DFE6    DFF8  BNE DFFE    (pour la fonction VAL seulement)
DFE2  STX D6      DFFA  STX D6      oui, signe négatif (#D6=#FF)
DFE4  BEQ DFEA    DFFC  BEQ E002    et continuer pour caractère suivant
DFE6  CMP #'+'    DFFE  CMP #'+'    est-ce un plus ?
DFE8  BNE DFEF    E000  BNE E007    oui, ignorer simplement
DFEA  JSR $00E2   E002  JSR $00E2   prendre caractère suivant
DFED  BCC E05E    E005  BCC E062    et sauter si chiffre
```

```
DFEF  CMP #'.'    E007  CMP #'.'    est-ce le point décimal ?
DFF1  BEQ E02B    E009  BEQ E039    oui, sauter
DFF3  CMP #'E'    E00B  CMP #'E'    est-ce l'indicateur d'exposant ?
DFF5  BNE E03B    E00D  BNE E03F    non, fin de l'évaluation

Evaluer l'exposant

DFF7  JSR $00E2   E00F  JSR $00E2   prendre premier caractère de l'exposant
DFFA  BCC E01D    E012  BCC E02B    sauter si chiffre
DFFC  CMP #&-     E014  CMP #&-     l'exposant est-il négatif ?
DFFE  BEQ E000    E016  BEQ E026    oui, l'indiquer
E000  CMP #'-'    E018  CMP #'-'    idem pour VAL
E002  BEQ E00E    E01A  BEQ E026    et même traitement...
E004  CMP #&+     E01C  CMP #&+     l'exposant est-il signalé explicitement + ?
E006  BEQ E01A    E01E  BEQ E028    oui, ignorer simplement
E008  CMP #'+'    E020  CMP #'+'    idem pour VAL
E00A  BEQ E01A    E022  BEQ E028    et même ignorance (!)
E00C  BNE E01F    E024  BNE E02D    fin du nombre: ajuster et sortir

E00E  LDA #00     .....             exposant négatif:
E010  BCC E014    .....
E012  LDA #80     .....
E014  LSR CF      .....             quelques octets pour mettre à un (C=1)
E016  ORA CF      .....             le Bit 7 de #CF
E018  STA CF      .....             tout à fait équivalent à la ligne suivante:
.....             E026  ROR CF      indiquer exposant négatif: #CF<0
E01A  JSR $00E2   E028  JSR $00E2   prendre caractère suivant
E01D  BCC E005    E02B  BCC E009    et sauter si chiffre sinon, fin du nombre
E01F  BIT CF      E02D  BIT CF      tester signe de l'exposant
E021  BPL E03B    E02F  BPL E03F    si positif,aller ajuster le nombre et finir
E023  LDA #00     E031  LDA #00     si négatif,
E025  SEC         E033  SEC         complémenter l'exposant
E026  SBC CD      E034  SBC CD      dans A
E028  JMP $E03D   E036  JMP $E041   et ajuster le nombre et finir aussi.

Traiter le point décimal

E02B  LDA #00     .....             toujours la même maestria
E02D  BCC E031    .....
E02F  LDA #80     .....             pour répercuter C dans
E031  LSR CE      .....
E033  ORA CE      .....             le Bit 7 de #CE
E035  STA CE      .....             qui dit qu'on est dans la partie décimale
.....             E039  ROR CE      c'est tellement plus simple comme ça...
```

```
E037  BIT CE      E03B  BIT CE
E039  BVC DFEA    E03D  BVC E002    si c'est la deuxième virgule, c'est fini
E03B  LDA CD      E03F  LDA CD      prendre exposant (base 10)
E03D  SEC         E041  SEC
E03E  SBC CC      E042  SBC CC      moins le nombre de chiffre après la virgule
E040  STA CD      E044  STA CD      comme exposant base 10
E042  BEQ E056    E046  BEQ E05A   si l'exposant est nul, c'est bon
E044  BPL E04F    E048  BPL E053   si il est positif, ajuster
```

Ajuster le nombre pour un exposant négatif

```
E046  JSR $DDBF   E04A  JSR $DDC3  ACC1=ACC1/10
E049  INC CD      E04D  INC CD     incrémenter l'exposant
E04B  BNE E046    E04F  BNE E04A   et recommencer tant que pas correct
E04D  BEQ E056    E051  BEQ E05A   complémenter si nombre négatif et finir
```

Ajuster le nombre pour un exposant positif

```
E04F  JSR $DDA3   E053  JSR $DDA7  ACC1=ACC1*10
E052  DEC CD      E056  DEC CD     décrémenter l'exposant
E054  BNE E04F    E058  BNE E053   et recommencer tant que pas correct
E056  LDA D6      E05A  LDA D6     prendre signe
E058  BMI E05B    E05C  BMI E05F   et sauter si négatif,
E05A  RTS         E05E  RTS        sortir sinon
E05B  JMP $E26B   E05F  JMP $E271  appliquer l'opérateur '-'

E05E  PHA         E062  PHA        sauver le chiffre (ASCII)
E05F  BIT CE      E063  BIT CE     tester si décimal ou entier
E061  BPL E065    E065  BPL E069   sauter si entier
E063  INC CC      E067  INC CC     incrémenter le nombre de chiffre décimaux
E065  JSR $DDA3   E069  JSR $DDA7  ACC1=ACC1*10
E068  PLA         E06C  PLA        récupérer chiffre (ASCII)
E069  SEC         E06D  SEC
E06A  SBC #30     E06E  SBC #30    et le ramener à 0-9
E06C  JSR $E072   E070  JSR $E076  et l'ajouter à ACC1
E06F  JMP $DFEA   E073  JMP $E002  puis continuer chiffre suivant
```

ACC1+A --) ACC1

```
E072  PHA         E076  PHA        sauver la valeur à ajouter sur la pile
E073  JSR $DEDD   E077  JSR $DEE5  AACC1 -) ACC2
E076  PLA         E07A  PLA        récupérer valeur
E077  JSR $DF15   E07B  JSR $DF24  et la convertir dans l'ACC1
```

```
E07A  LDA DD      E07E  LDA DD      prendre signe ACC2
E07C  EOR D5      E080  EOR D5      et faire le OU exclusif avec celui de ACC1
E07E  STA DE      E082  STA DE      le sauver en #DE
E080  LDX D0      E084  LDX D0      Positionner Z si ACC1 nul
E082  JMP $DA9A   E086  JMP $DB25   effectuer ACC1=ACC1+ACC2
```

Détermination de l'exposant

Entrée: TXTPTR pointe sur le caractère de l'exposant

Sortie: #CD contient la nouvelle valeur

Remarque: si l'exposant est déjà supérieur à 10, c'est que l'exposant a trois chiffres, ce qui veut dire qu'il y a overflow s'il est positif.
S'il est négatif, il peut avoir autant de chiffre qu'il veut, l'exposant sera de toutes façons à -100, ce qui veut dire que le nombre est 0

```
E085  LDA CD      E089  LDA CD      prendre exposant courant
E087  CMP #0A     E08B  CMP #0A
E089  BCC E094    E08D  BCC E098    et sauter si inférieur à 10
E08B  LDA #64     E08F  LDA #64     préparer pour exposant =100
E08D  BIT CF      E091  BIT CF      prendre signe de l'exposant
E08F  BMI E0A2    E093  BMI E0A6    OK si négatif
E091  JMP $DBE0   E095  JMP $DC39   'OVERFLOW ERROR' sinon
E094  ASL A       E098  ASL A       *2
E095  ASL A       E099  ASL A       *4
E096  CLC         E09A  CLC         +ancien:
E097  ADC CD      E09B  ADC CD      *5
E099  ASL A       E09D  ASL A       *10
E09A  CLC         E09E  CLC         (c'était donc le chiffre des dizaines)
E09B  LDY #00     E09F  LDY #00
E09D  ADC (E9),Y  E0A1  ADC (E9),Y  et ajouter deuxième chiffre de l'exposant
E09F  SEC         E0A3  SEC
E0A0  SBC #30     E0A4  SBC #30     rajuster à cause du code ASCII
E0A2  STA CD      E0A6  STA CD      et sauver l'exposant à nouveau
E0A4  JMP $E01A   E0A8  JMP $E028   et continuer exposant
```

CONSTANTES POUR CONVERSION FLOTTANT/DECIMAL

```
E0A7 E0AB  BYT #9B,#3E,#BC,#1F,#FD  soit 999 999.9
E0AC E0B0  BYT #9E,#6E,#6B,#27,#FD  soit 999 999 999
E0B1 E0B5  BYT #9E,#6E,#6B,#28,#00  soit 1 000 000 000
```

AFFICHER 'IN ' + No DE LIGNE

```
E0B6  LDA #B1     E0BA  LDA #AD     AY pointe sur
E0B8  LDY #C3     E0BC  LDY #C3     le message 'IN '
E0BA  JSR $E0CE   E0BE  JSR $E0D2   et l'afficher
E0BD  LDA A9      E0C1  LDA A9      prendre dans XA
E0BF  LDX A8      E0C3  LDX A8      le Numéro de la ligne courante
```

AFFICHER AX COMME ENTIER NON SIGNE

```
E0C1  STA D1      E0C5  STA D1      sauver poids fort
E0C3  STX D2      E0C7  STX D2      et poids faible
E0C5  LDX #90     E0C9  LDX #90     indiquer exposant (base 2) =16
E0C7  SEC         E0CB  SEC         indiquer positif (pas de complémentation)
E0C8  JSR $DF22   E0CC  JSR $DF31   et convertir en flottant (ou... #D8D5/DF40)
E0CB  JSR $E0D1   E0CF  JSR $E0D5   puis en décimal
E0CE  JMP $CBED   E0D2  JMP $CCB0   et enfin afficher la chaine décimale
```

CONVERTIR ACC1 EN DECIMAL

Entrée: le nombre est dans ACC1.

Sortie: AY=adresse de la chaine (#0100), le nombre est converti en décimal à cette adresse, terminé par le NULL (#00). ACC1 est détruit.

Principe: assez compliqué. Dans un premier temps,il va falloir déterminer si on doit utiliser ou non la notation scientifique, et si non, calculer la place de la virgule, en justifiant correctement le nombre pour afficher un maximum de chiffres significatifs. On effectue seulement alors la conversion en base 10, décrite plus loin.*

Bogue: V1.0: si le nombre est positif, son signe est #02 au lieu de #20 (espace)

```
E0D1  LDY #01     E0D5  LDY #01     pointer sur le début du tampon décimal
```

Placer signe et test si nombre nul

```
E0D3  LDA #02     E0D7  LDA #20     préparer A pour signe + ,bogue pour Oric 1
E0D5  BIT D5      E0D9  BIT D5      prendre signe de ACC1
E0D7  BPL E0DB    E0DB  BPL E0DF    si positif, OK
E0D9  LDA #'-'    E0DD  LDA #'-'    si négatif, prendre '-'
```

```
E0DB  STA 00FF,Y  E0DF  STA 00FF,Y  et sauver le signe dans le tampon
E0DE  STA D5      E0E2  STA D5      et aussi dans signe ACC1
E0E0  STY E0      E0E4  STY E0      sauver le pointeur de tampon
E0E2  INY         E0E6  INY         pointer sur le premier chiffre
E0E3  LDA #'0'    E0E7  LDA #'0'    préparer A si nombre nul
E0E5  LDX D0      E0E9  LDX D0      prendre exposant
E0E7  BNE E0EC    E0EB  BNE E0F0    sauter si ACC1 <=> 0
E0E9  JMP $E1F4   E0ED  JMP $E1F8   si ACC1=0, mettre 0 et sortir

Ajuster ACC1 pour précision maximum

E0EC  LDA #00     E0F0  LDA #00
E0EE  CPX #80     E0F2  CPX #80     tester exposant (base 2)
E0F0  BEQ E0F4    E0F4  BEQ E0F8    sauter si le nombre est de l'ordre de 1
E0F2  BCS E0FD    E0F6  BCS E101    sauter s'il est plus grand que 1
E0F4  LDA #B1     E0F8  LDA #B5     indexer 1 000 000 000 (1 E+10)
E0F6  LDY #E0     E0FA  LDY #E0     la plus grande valeur exclue sans exposant
E0F8  JSR $DCB7   E0FC  JSR $DCED   ACC1*(AY) --> ACC1
E0FB  LDA #F7     E0FF  LDA #F7     exposant base 10=-10 pour compenser
E0FD  STA CC      E101  STA CC      la multiplication par 1 E+10
E0FF  LDA #AC     E103  LDA #B0     indexer 999 999 999 (1E+10 -1)
E101  LDY #E0     E105  LDY #E0
E103  JSR $DF34   E107  JSR $DF4C   et comparer à ACC1
E106  BEQ E126    E10A  BEQ E12A    si égal, précision maxi, on saute
E108  BPL E11C    E10C  BPL E120    si plus grand, il faut diviser
E10A  LDA #A7     E10E  LDA #AB     si plus petit, on encadre
E10C  LDY #E0     E110  LDY #E0     avec la valeur 999 999.9 (1E+07 -1)
E10E  JSR $DF34   E112  JSR $DF4C
E111  BEQ E115    E115  BEQ E119    si c'est égal, il faut multiplier
E113  BPL E123    E117  BPL E127    si ACC1 plus grand, plus d'ajustement

E115  JSR $DDA3   E119  JSR $DDA7   ACC1=ACC1*10
E118  DEC CC      E11C  DEC CC      et compenser avec puissance de 10
E11A  BNE E10A    E11E  BNE E10E    inconditionnel: continuer ajustement

E11C  JSR $DDBF   E120  JSR $DDC3   ACC1=ACC1/10
E11F  INC CC      E123  INC CC      et compenser avec puissance de 10
E121  BNE E0FF    E125  BNE E103    inconditionnel: continuer ajustement

Tester notation scientifique, placer la virgule

E123  JSR $DA79   E127  JSR $DB04   ACC1+0,5 --> ACC1 (arrondi base 10)
E126  JSR $DF74   E12A  JSR $DF8C   ACC1 --> #D4-#D3-#D2-#D1
E129  LDX #01     E12D  LDX #01     initaliser point décimal
```

```
E12B  LDA CC      E12F  LDA CC      prendre exposant base 10
E12D  CLC         E131  CLC         on ajoute 10 (1 E+10), c'est le maximum
E12E  ADC #0A     E132  ADC #0A     que l'on puisse afficher sans la notation
E130  BMI E13B    E134  BMI E13F    scientifique. si <0, notation scientifique
E132  CMP #0B     E136  CMP #0B     exposant supérieur à 10 ?
E134  BCS E13C    E138  BCS E140    oui, notation scientifique
E136  ADC #FF     E13A  ADC #FF     non, calculer la place de la virgule
E138  TAX         E13C  TAX         et la sauver
E139  LDA #02     E13D  LDA #02
E13B  SEC         E13F  SEC
E13C  SBC #02     E140  SBC #02
E13E  STA CD      E142  STA CD
E140  STX CC      E144  STX CC      sauver position de la virgule
E142  TXA         E146  TXA
E143  BEQ E147    E147  BEQ E14B    si 0, on affiche tout de suite
E145  BPL E15A    E149  BPL E15E    si >0, pas de virgule
E147  LDY E0      E14B  LDY E0      prendre position dans le tampon
E149  LDA #'.'    E14D  LDA #'.'    et code pour virgule (point décimal)
E14B  INY         E14F  INY         on ajuste l'index
E14C  STA 00FF,Y  E150  STA 00FF,Y  et on écrit la virgule
E14F  TXA         E153  TXA         si uniquement partie décimale
E150  BEQ E158    E154  BEQ E15C
E152  LDA #'0'    E156  LDA #'0'    , on place un '0' devant la virgule
E154  INY         E158  INY
E155  STA 00FF,Y  E159  STA 00FF,Y
```

Conversion en décimal

Principe: simple, mais l'optimisation de la routine complique pas mal les choses.

Le principe général est classique (et obligatoire ou presque): soustraire du nombre les puissances décroissantes de 10, jusqu'à ce que le nombre soit nul. Mais cette méthode, utilisée de manière classique nécessite une comparaison (le nombre est il inférieur à la puissance de dix courante ?) suivie éventuellement de la soustraction.

Pour éviter cette comparaison, une astuce a été utilisée: utilisée des puissances de 10 alternativement positives et négatives, ce qui permet de se plonger dans les délices des calculs avec des nombres signés...

Voici un exemple de la méthode, pour convertir le nombre 2103 (que l'on imaginera stocké en binaire signé) en décimal.

Opération reste résultat

-1000 1103 0

```
-1000     103  1
-1000   - 897  2
+ 100   - 797  20
+ 100   - 697  21
     ....
+ 100   -  97  27
+ 100      03  28 --> 21 (complément à 10)
-  10   -   7  210
+   1   -   6  2100
+   1   -   5  2101
     ....
+   1   -   1  2105
+   1       0  2106 --> 2103 (complément à 10)
```

```
E158  STY E0      E15C  STY E0      sauver position dans le tampon

E15A  LDY #00     E15E  LDY #00     indexer début de la table
E15C  LDX #80     E160  LDX #80     indiquer chiffre=0 pour l'instant
E15E  LDA D4      E162  LDA D4      ajouter mantisse et valeur de la table
E160  CLC         E164  CLC
E161  ADC E209,Y  E165  ADC E20D,Y  octet 4
E164  STA D4      E168  STA D4
E166  LDA D3      E16A  LDA D3
E168  ADC E208,Y  E16C  ADC E20C,Y  octet 3
E16B  STA D3      E16F  STA D3
E16D  LDA D2      E171  LDA D2
E16F  ADC E207,Y  E173  ADC E20B,Y  octet 2
E172  STA D2      E176  STA D2
E174  LDA D1      E178  LDA D1
E176  ADC E206,Y  E17A  ADC E20A,Y  octet 1
E179  STA D1      E17D  STA D1
E17B  INX         E17F  INX
E17C  BCS E182    E180  BCS E186    sauter si dépassement
E17E  BPL E15E    E182  BPL E162    phase 'addition', on continue
E180  BMI E184    E184  BMI E188    sinon, on sort
E182  BMI E15E    E186  BMI E162    phase 'soustraction' on continue
E184  TXA         E188  TXA
E185  BCC E18B    E189  BCC E18F    phase soustraction: sauter complémentation
E187  EOR #FF     E18B  EOR #FF     complémenter à 10 (C=1)
E189  ADC #0A     E18D  ADC #0A
E18B  ADC #2F     E18F  ADC #2F     convertir en ASCII
E18D  INY         E191  INY
E18E  INY         E192  INY
E18F  INY         E193  INY
```

```
E19Ø  INY          E194  INY          indexer élément suivant dans la table
E191  STY B6       E195  STY B6       et sauver index table
E193  LDY EØ       E197  LDY EØ       reprendre index tampon
E195  INY          E199  INY          et ajuster suivant
E196  TAX          E19A  TAX          sauver code
E197  AND #7F      E19B  AND #7F      éliminer b7
E199  STA ØØFF,Y   E19D  STA ØØFF,Y   et placer dans tampon
E19C  DEC CC       E1AØ  DEC CC       et calculer si point décimal à placer
E19E  BNE E1A6     E1A2  BNE E1AA     non,on saute
E1AØ  LDA #'.'     E1A4  LDA #'.'
E1A2  INY          E1A6  INY
E1A3  STA ØØFF,Y   E1A7  STA ØØFF,Y   oui,on la place
E1A6  STY EØ       E1AA  STY EØ       sauver index tampon de sortie
E1A8  LDY B6       E1AC  LDY B6       récupérer index addition
E1AA  TXA          E1AE  TXA          calculer indicateur
E1AB  EOR #FF      E1AF  EOR #FF
E1AD  AND #8Ø      E1B1  AND #8Ø
E1AF  TAX          E1B3  TAX          et le remettre dans X
E1BØ  CPY #24      E1B4  CPY #24      fin de la conversion ?
E1B2  BNE E15E     E1B6  BNE E162     non, on repart

Eliminer les Ø après la virgule

E1B4  LDY EØ       E1B8  LDY EØ       prendre index du tampon
E1B6  LDA ØØFF,Y   E1BA  LDA ØØFF,Y   et dernier caractère courant
E1B9  DEY          E1BD  DEY          indexer précédent
E1BA  CMP #'Ø'     E1BE  CMP #'Ø'     est-ce un 'Ø' ?
E1BC  BEQ E1B6     E1CØ  BEQ E1BA     oui,on ignore et on recommence
E1BE  CMP #'.'     E1C2  CMP #'.'     si c'est une virgule,
E1CØ  BEQ E1C3     E1C4  BEQ E1C7     on ignore mais on sort (Y a été décrémenté)
E1C2  INY          E1C6  INY          sinon,rajuster Y sur position libre

Placer l'exposant

E1C3  LDA #'+'     E1C7  LDA #'+'     exposant positif
E1C5  LDX CD       E1C9  LDX CD       prendre exposant base 1Ø
E1C7  BEQ E1F7     E1CB  BEQ E1FB     si Ø, rien à faire, on sort
E1C9  BPL E1D3     E1CD  BPL E1D7     si positif,sauver le signe
E1CB  LDA #ØØ      E1CF  LDA #ØØ      exposant négatif:
E1CD  SEC          E1D1  SEC          on le complémente
E1CE  SBC CD       E1D2  SBC CD
E1DØ  TAX          E1D4  TAX          et le remet dans X
E1D1  LDA #'-'     E1D5  LDA #'-'     indiquer exposant négatif
E1D3  STA Ø1Ø1,Y   E1D7  STA Ø1Ø1,Y
```

```
mettre le signe un caractère plus loin
E1D6  LDA #'E'    E1DA  LDA #'E'    puis indicateur exposant
E1D8  STA Ø1ØØ,Y  E1DC  STA Ø1ØØ,Y

E1DB  TXA         E1DF  TXA         exposant dans A
E1DC  LDX #2F     E1EØ  LDX #2F     préparer code ASCII
E1DE  SEC         E1E2  SEC
E1DF  INY         E1E3  INX         code ASCII+1
E1EØ  SBC #ØA     E1E4  SBC #ØA     on enlève 1Ø
E1E2  BCS E1DF    E1E6  BCS E1E3    si toujours des dizaines, on recommence
E1E4  ADC #3A     E1E8  ADC #3A     on met le 2ème chiffre (#3A='Ø'+1Ø)
E1E6  STA Ø1Ø3,Y  E1EA  STA Ø1Ø3,Y  mettre chiffre des unités
E1E9  TXA         E1ED  TXA
E1EA  STA Ø1Ø2,Y  E1EE  STA Ø1Ø2,Y  puis chiffre des dizaines
E1ED  LDA #ØØ     E1F1  LDA #ØØ     terminer la chaine
E1EF  STA Ø1Ø4,Y  E1F3  STA Ø1Ø4,Y  par un NULL
E1F2  BEQ E1FC    E1F6  BEQ E2ØØ    inconditionnel:finir

E1F4  STA ØØFF,Y  E1F8  STA ØØFF,Y  sauver le 'Ø' du nombre nul
E1F7  LDA #ØØ     E1FB  LDA #ØØ
E1F9  STA Ø1ØØ,Y  E1FD  STA Ø1ØØ,Y  et le faire suivre d'un Ø (NULL)
E1FC  LDA #ØØ     E2ØØ  LDA #ØØ
E1FE  LDY #Ø1     E2Ø2  LDY #Ø1     AY pointe sur #1ØØ,début de la chaine
E2ØØ  RTS         E2Ø4  RTS
```

CONSTANTE POUR ARRONDIR,SGR ETC...

```
E2Ø1 E2Ø5  BYT #8Ø,#ØØ,#ØØ,#ØØ,#ØØ  soit Ø,5
```

CONSTANTES POUR CONVERSION EN DECIMAL

Ces constantes sont stockées sur 4 octets signés, c'est à dire en complément à 2 pour les constantes négatives.

```
E2Ø6 E2ØA  BYT #FA,#ØA,#1F,#ØØ      soit -1ØØ ØØØ ØØØ
E2ØA E2ØE  BYT #ØØ,#98,#96,#8Ø      soit   1Ø ØØØ ØØØ
E2ØE E212  BYT #FF,#FØ,#BD,#CØ      soit   -1 ØØØ ØØØ
E212 E216  BYT #ØØ,#Ø1,#86,#AØ      soit      1ØØ ØØØ
E216 E21A  BYT #FF,#FF,#D8,#FØ      soit      -1Ø ØØØ
E21A E21E  BYT #ØØ,#ØØ,#Ø3,#E8      soit        1 ØØØ
E21E E222  BYT #FF,#FF,#FF,#9C      soit         -1ØØ
E222 E226  BYT #ØØ,#ØØ,#ØØ,#ØA      soit           1Ø
E226 E22A  BYT #FF,#FF,#FF,#FF      soit           -1
```

'SQR' (FONCTION) SQR(ACC1) --> ACC1

Principe: SQR(X)=X^(1/2)

```
E22A  JSR $DEDD    E22E  JSR $DEE5    AACC1 --> ACC2
E22D  LDA #01      E231  LDA #05
E22F  LDY #E2      E233  LDY #E2      indexer 1/2
```

ACC2^(AY) --> ACC1

```
E231  JSR $DE73    E235  JSR $DE7B    (AY) --> ACC1
```

'^' (OPERATEUR) ACC2^ACC1 --> ACC1

Principe:X^Y=EXP(Y*LN(X))

Si Y=0,X^Y=1

Si X=0,X^Y=0

L'exponentiation est donc un opérateur très lent, puisqu'il nécessite une multiplication, le calcul d'un Log et enfin une exponentielle. La précision s'en ressent naturellement:2^9 donne 512,000001 !

Attention: X et Y désignent ici les deux opérandes.

```
E234  BEQ E2A6     E238  BEQ E2AA    si Y=0, calculer directement EXP(0)
E236  LDA D8       E23A  LDA D8      X=0 ?
E238  BNE E23D     E23C  BNE E241    non, OK
E23A  JMP $DB29    E23E  JMP $DBB4   oui, alors résultat nul
E23D  LDX #BD      E241  LDX #BD
E23F  LDY #00      E243  LDY #00      indexer #00BD
E241  JSR $DEA5    E245  JSR $DEAD    et y sauver ACC1 (Y)
E244  LDA DD       E248  LDA DD       prendre signe de X
E246  BPL E257     E24A  BPL E25B     positif, c'est bon
E248  JSR $DFA5    E24C  JSR $DFBD    calculer INT(Y)
E24B  LDA #BD      E24F  LDA #BD
E24D  LDY #00      E251  LDY #00
E24F  JSR $DF34    E253  JSR $DF4C    et comparer à Y
E252  BNE E257     E256  BNE E25B     si différent, Y non entier, on saute
E254  TYA          E258  TYA          A=0-4, indiquer signe positif
E255  LDY 24       E259  LDY 24       poids faible dans Y (b0 est donc la parité)
E257  JSR $DECF    E25B  JSR $DED7    ACC2 --> ACC1 (X --> ACC1) (signe dans A)
E25A  TYA          E25E  TYA
E25B  PHA          E25F  PHA          sauver parité
```

```
E25C  JSR $DC79   E260  JSR $DCAF   calcul de LN(X) dans ACC1
E25F  LDA #BD     E263  LDA #BD
E261  LDY #00     E265  LDY #00     indexer #00BD (où est stocké Y)
E263  JSR $DCB7   E267  JSR $DCE0   (AY)↑ACC1 --) ACC1
E266  JSR $E2A6   E26A  JSR $E2AA   EXP(ACC1) --) ACC1,soit EXP(Y↑LN(X))
E269  PLA         E26D  PLA         récupérer parité de X
E26A  LSR A       E26E  LSR A       dans C
E26B  BCC E277    E26F  BCC E27B   si pair, résultat positif
```

'-' (OPERATEUR DE CHANGEMENT DE SIGNE)

```
E26D  LDA D0      E271  LDA D0      si le nombre est nul, on sort
E26F  BEQ E277    E273  BEQ E27B
E271  LDA D5      E275  LDA D5      sinon, inverser le signe
E273  EOR #FF     E277  EOR #FF
E275  STA D5      E279  STA D5
E277  RTS         E27B  RTS

E278 E27C  BYT #81,#38,#AA,#3B,#29  soit  1,44269504  ou encore 1/LN(2)
```

DONNEES DU POLYNOME DE CALCUL DE EXP(X)

```
E27D E281  BYT #07
E27E E282  BYT #71,#34,#58,#3E,#56  soit  0,0000214987637 Coéfficient:A7
E283 E287  BYT #74,#16,#7E,#B3,#1B  soit  0,0001435231400 Coéfficient:A6
E288 E28C  BYT #77,#2F,#EE,#E3,#85  soit  0,0013422634800 Coéfficient:A5
E28D E291  BYT #7A,#1D,#84,#1C,#2A  soit  0,0096140170100 Coéfficient:A4
E292 E296  BYT #7C,#63,#59,#58,#0A  soit  0,0555051269    Coéfficient:A3
E297 E29B  BYT #7E,#75,#FD,#E7,#C6  soit  0,2402263850    Coéfficient:A2
E29C E2A0  BYT #80,#31,#72,#18,#10  soit  0,6931471860    Coéfficient:A1
E2A1 E2A5  BYT #81,#00,#00,#00,#00  soit  1               Coéfficient:A0
```

'EXP' (FONCTION) EXP(ACC1) --) ACC1

Principe: se ramener au calcul dans l'intervalle 0,1 a nécesité une astuce:

Figure E2-A

$e^x = e^{\ln(2).(x / \ln(2))} = (e^{\ln(2)})^{(x / \ln(2))}$

Or, $e^{\ln(2)} = 2$

on obtient donc $e^x = 2^{x / \ln(2)}$

posons $x / \ln(2) = y$

nous pouvons séparer y en partie entière et décimale:

$y = E + D$

d'où $e^x = 2^{(E+D)} = 2^E.2^D$

Etant donné le codage en virgule flottante, multiplier un nombre par 2^E revient à ajouter E à son exposant.

Il suffit donc de calculer 2^D, $0 <= D < 1$

Le D.L de e^x est $1 + x + x^2 / 2 + ... + x^n / n!$

donc, puisque $x=y.\ln(2)$: $e^x = 1 + \ln(x).y + ... + (\ln(2)^n / n!) y^n$

Le terme général du polynome est donc voisin de $\ln(2)^n / n!$

```
E2A6  LDA #78     E2AA  LDA #7C
E2A8  LDY #E2     E2AC  LDY #E2     indexer 1/LN(2)
E2AA  JSR $DCB7   E2AE  JSR $DCED   (AY)*ACC1 --) ACC1,soit X/LN(2)
E2AD  LDA DF      E2B1  LDA DF      prendre extension
E2AF  ADC #50     E2B3  ADC #50
E2B1  BCC E2B6    E2B5  BCC E2BA
E2B3  JSR $DEF4   E2B7  JSR $DEFC   arrondir d'après l'extension
E2B6  STA C5      E2BA  STA C5      sauver nouvelle extension
E2B8  JSR $DEE0   E2BC  JSR $DEE8   ACC1 --) ACC2
E2BB  LDA D0      E2BF  LDA D0      prendre exposant
E2BD  CMP #88     E2C1  CMP #88     supérieur à 8 ?
E2BF  BCC E2C4    E2C3  BCC E2C8    non, sauter
E2C1  JSR $DD95   E2C5  JSR $DD99   si nombre négatif: ACC1=0, sinon overflow
E2C4  JSR $DFA5   E2C8  JSR $DFBD   INT(ACC1) --) ACC1
E2C7  LDA 24      E2CB  LDA 24      prendre poids faible (en fait le nombre)
E2C9  CLC         E2CD  CLC
E2CA  ADC #81     E2CE  ADC #81     +129
E2CC  BEQ E2C1    E2D0  BEQ E2C5    si égal à 127, 0 ou overflow
```

```
E2CE  SEC          E2D2  SEC          (tout ceci pour convertir en exposant)
E2CF  SBC #Ø1      E2D3  SBC #Ø1      -1=+128
E2D1  PHA          E2D5  PHA          et on sauve
E2D2  LDX #Ø5      E2D6  LDX #Ø5
E2D4  LDA D8,X     E2D8  LDA D8,X
E2D6  LDY DØ,X     E2DA  LDY DØ,X     échanger ACC2 et ACC1
E2D8  STA DØ,X     E2DC  STA DØ,X     ACC1=X/LN(2)
E2DA  STY D8,X     E2DE  STY D8,X     ACC2=INT(X/LN(2))
E2DC  DEX          E2EØ  DEX
E2DD  BPL E2D4     E2E1  BPL E2D8
E2DF  LDA C5       E2E3  LDA C5       récupérer extension ACC2
E2E1  STA DF       E2E5  STA DF       dans ACC1 aussi
E2E3  JSR $DA83    E2E7  JSR $DBØE    ACC2-ACC1 --) ACC1
E2E6  JSR $E26D    E2EA  JSR $E271    -ACC1 --) ACC1 (X/LN(2)-INT(X/LN(2))
E2E9  LDA #7D      E2ED  LDA #81      ou encore calcul de la partie décimale
E2EB  LDY #E2      E2EF  LDY #E2      indexer polynome
E2ED  JSR $E3ØF    E2F1  JSR $E313    et calculer
E2FØ  LDA #ØØ      E2F4  LDA #ØØ      produit des signe positif
E2F2  STA DE       E2F6  STA DE
E2F4  PLA          E2F8  PLA          récupérer exposant à ajouter
E2F5  JSR $DD7A    E2F9  JSR $DD7E    ajouter A à l'exposant de ACC1
E2F8  RTS          E2FC  RTS
```

ACC1*P(ACC1*ACC1) --) ACC1

Principe: les fonctions COS ou SINUS par exemple ont des D.L qui s'expriment selon des puissances de l'ordre de 1,3,5,7 etc.. Pour éviter de calculer un polynome normal dont les termes de degré pair seraient nuls, on calcule un polynome du carré du nombre, on obtient donc un polynome de puissances paires (Ø,2,4,6 etc...). Il suffit de multiplier le résultat par le nombre ppour obtenir un polynome de degrés impairs.

```
E2F9  STA EØ       E2FD  STA EØ
E2FB  STY E1       E2FF  STY E1       sauver adresse du polynome
E2FD  JSR $DE9B    E3Ø1  JSR $DEA3    ACC1 --) #C6-#CA:sauver ACC1 pour future *
E3ØØ  LDA #C6      E3Ø4  LDA #C6      AY=#ØØC^
E3Ø2  JSR $DCB7    E3Ø6  JSR $DCED    (AY)*ACC1 --) ACC1,soit X^2
E3Ø5  JSR $E313    E3Ø9  JSR $E317    calculer le polynome,soit P(X^2)
E3Ø8  LDA #C6      E3ØC  LDA #C6
E3ØA  LDY #ØØ      E3ØE  LDY #ØØ      indexer valeur de X sauvée
E3ØC  JMP $DCB7    E31Ø  JMP $DCED    et calculer X*P(X^2)
```

P(ACC1) --) ACC1

Principe: si on appelle X le nombre dont on veut calculer le polynome, et A0, A1... les coefficients de la table, on calcule le polynome, s'il y a 6 coefficients

P(X)=(((((A5*X+A4)*X+A3)*X+A2)*X+A1)*X+A0)

Qui, développé, donne: A0+A1*X+A2*X^2+A3*X^3+A4*X^4+A5*X^5

La forme parenthésée évite de calculer les puissances successives de X, ce qui prendrait beaucoup de temps.

```
E30F  STA E0      E313  STA E0
E311  STY E1      E315  STY E1     sauver adresse du polynome
E313  JSR $DE98   E317  JSR $DEA0  ACC1 --) #CB-#CF
E316  LDA (E0),Y  E31A  LDA (E0),Y prendre nombre de valeurs
E318  STA D6      E31C  STA D6     et sauver pour compteur
E31A  LDY E0      E31E  LDY E0     prendre poids faible
E31C  INY         E320  INY        et ajuster sur 1er nombre
E31D  TYA         E321  TYA        dans A
E31E  BNE E322    E322  BNE E326
E320  INC E1      E324  INC E1     ajuster poids fort éventuellement
E322  STA E0      E326  STA E0     auver poids faible courant
E324  LDY E1      E328  LDY E1     et AY=adresse du coéfficient courant
E326  JSR $DCB7   E32A  JSR $DCED  et multiplier ACC1 par le coefficient
E329  LDA E0      E32D  LDA E0
E32B  LDY E1      E32F  LDY E1     récupérer dans AY adresse coefficient
E32D  CLC         E331  CLC
E32E  ADC #05     E332  ADC #05    passer au suivant
E330  BCC E333    E334  BCC E337
E332  INY         E336  INY        poids fort aussi
E333  STA E0      E337  STA E0
E335  STY E1      E339  STY E1     et resauver
E337  JSR $DA97   E33B  JSR $DB22  et ajouter ACC1 au nouveau coefficient
E33A  LDA #CB     E33E  LDA #CB
E33C  LDY #00     E340  LDY #00    indexer X original
E33E  DEC D6      E342  DEC D6     encore des coéficients ?
E340  BNE E326    E344  BNE E32A   oui,on recommence
E342  RTS         E346  RTS
```

VALEUR DE LA SUITE DU RND

```
E343 E347  BYT #98,#35,#44,#7A,#6B  soit 11879546,4
E348 E34C  BYT #68,#28,#B1,#46,#20  soit 3,9276777 E-08
```

'RND' (FONCTION)

Principe: on utilise une suite récurrente Un+1=a*Un+b, à laquelle on fait subir quelques outrages (permutation d'octets de la mantisse) et dont on ramène l'exposant à Ø, pour avoir un nombre entre Ø et 1. La formule résultante est très complexe, disons simplement qu'elle donne satisfaction !

```
E34B  JSR $DFØ4    E34F  JSR $DF13   A=SGN(ACC1)
E34E  TAX          E352  TAX         sauver dans X
E34F  BMI E369     E353  BMI E36D    si <Ø,prendre comme base
E351  LDA #FA      E355  LDA #FA     si >=Ø
E353  LDY #ØØ      E357  LDY #ØØ     indexer valeur courante
E355  JSR $DE73    E359  JSR $DE7B   et la mettre dans ACC1
E358  TXA          E35C  TXA         l'argument était nul ?
E359  BEQ E342     E35D  BEQ E346    oui, on sort
E35B  LDA #43      E35F  LDA #47
E35D  LDY #E3      E361  LDY #E3     non, indexer valeur pour multiplication
E35F  JSR $DCB7    E363  JSR $DCED   et faire la multiplication
E362  LDA #47      E366  LDA #4B
E364  LDY #E3      E368  LDY #E3     indexer valeur pour addition
E366  JSR $DA97    E36A  JSR $DB22   et faire l'addition
E369  LDX D4       E36D  LDX D4
E36B  LDA D1       E36F  LDA D1
E36D  STA D4       E371  STA D4
E36F  STX D1       E373  STX D1      permuter octet 1 et 4
E371  LDA #ØØ      E375  LDA #ØØ
E373  STA D5       E377  STA D5      indiquer signe +
E375  LDA DØ       E37?  LDA DØ      exposant...
E377  STA DF       E37B  STA DF      comme extension
E379  LDA #8Ø      E37D  LDA #8Ø     et exposant
E37B  STA DØ       E37F  STA DØ      pour nombre <1
E37D  JSR $DBØ7    E381  JSR $DB92   ajuster la mantisse
E38Ø  LDX #FA      E384  LDX #FA
E382  LDY #ØØ      E386  LDY #ØØ     indexer valeur courante
E384  JMP $DEA5    E388  JMP $DEAD   et y sauver la nouvelle valeur
```

'COS' (FONCTION)

Principe:On utilise l'égalité classique:COS(X)=SIN(X+PI/2)

```
E387  LDA #Ø3      E38B  LDA #Ø7
E389  LDY #E4      E38D  LDY #E4     indexer la valeur PI/2
E38B  JSR $DA97    E38F  JSR $DB22   et faire ACC1+PI/2
```

'SIN' (FONCTION)

Principe: pour avoir une convergence rapide, on commence par se ramener l'intervalle à Ø-1 en divisant le nombre par 2PI, et en prenant sa partie entière, ce qui permet aussi de prendre l'angle modulo 2 PI.

Toutefois, étant données les propriétés du cercle trigonométrique, il suffit de calculer le sinus entre Ø et PI/2 (donc entre Ø et 1/4). Le D.L étant calculé en 1, la précision pour des angles voisins de Ø est douteuse !

Figure E3-A

Le D.L normal de sin (x) est (attention, il est calculé en 1):

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots + (-1)^p \frac{x^{2p+1}}{(2p+1)!}$$

Or on calcule en fait le sin de $y = \frac{x}{2\pi}$ donc $x = 2\pi y$

$$\text{et } \sin(y) = 2\pi y + \frac{(2\pi y)^3}{3!} + \ldots + (-1)^p \frac{(2\pi y)^{2p+1}}{(2p+1)!}$$

le terme général est donc voisin de: $(-1)^p \frac{(2\pi)^{2p+1}}{(2p+1)!}$

Ramener x / 2π à l'intervalle 0,1/4, c'est ramener x à l'intervalle 0,π / 2

on obtient donc:

0 < x < π / 2	(0,1 / 4) :	sin (x) = sin (x)	ou sin (2πy) = sin (2πy)
π / 2 < x < π	(1 / 4, 1 / 2):	sin (x) = -sin (π+x)	ou sin (2πy) = -sin (2π(1/2+y))
π < x < 3π / 2	(1 / 2, 3 / 4):	sin (x) = -sin (π-x)	ou sin (2πy) = -sin(2π(1 / 2-y))
3π / 2 < x < 2π	(3 / 4, 1) :	sin (x) = -sin (x)	ou sin (2πy) = -sin (2πy)

```
E38E  JSR $DEDD   E392  JSR $DEE5   AACC1 --> ACC2
E391  LDA #Ø8     E395  LDA #ØC
E393  LDY #E4     E397  LDY #E4     indexer 2*PI
E395  LDX DD      E399  LDX DD      prendre signe de ACC2 comme produit signes
E397  JSR $DDC8   E39B  JSR $DDCC   et calculer X/(2*PI)
```

```
E39A  JSR $DEDD   E39E  JSR $DEE5   AACC1 --) ACC2
E39D  JSR $DFA5   E3A1  JSR $DFBD   INT(ACC1) --) ACC1
E3A0  LDA #00     E3A4  LDA #00
E3A2  STA DE      E3A6  STA DE      produit des signes positif
E3A4  JSR $DA83   E3A8  JSR $DB0E   et calculer ACC2-ACC1,i.e partie décimale
E3A7  LDA #0D     E3AB  LDA #11
E3A9  LDY #E4     E3AD  LDY #E4     indexer 1/4
E3AB  JSR $DA80   E3AF  JSR $DB0B   et faire 1/4-ACC1 (intervalle: -3/4 à 1/4)
E3AE  LDA D5      E3B2  LDA D5      prendre signe
E3B0  PHA         E3B4  PHA         et sauver
E3B1  BPL E3C0    E3B5  BPL E3C4    si positif (0 à 1/4 originel), sauter
E3B3  JSR $DA79   E3B7  JSR $DB04   ajouter 1/2, pour recentrer l'intervalle
E3B6  LDA D5      E3BA  LDA D5      (-1/4 à 1/2 donc) reprendre signe
E3B8  BMI E3C3    E3BC  BMI E3C7    négatif, c'est bon (3/4 à 1 originel)
E3BA  LDA 2D      E3BE  LDA 2D      si positif, (1/4 à 3/4 originel)
E3BC  EOR #FF     E3C0  EOR #FF     on inverse le signe
E3BE  STA 2D      E3C2  STA 2D
E3C0  JSR $E26D   E3C4  JSR $E271   dans ACC1 aussi
E3C3  LDA #0D     E3C7  LDA #11
E3C5  LDY #E4     E3C9  LDY #E4     on indexe  la valeur1/4
E3C7  JSR $DA97   E3CB  JSR $DB22   et on ajoute, on retrouve -1/4 à 1/4
E3CA  PLA         E3CE  PLA         récupérer signe
E3CB  BPL E3D0    E3CF  BPL E3D4    si positif, c'est bon
E3CD  JSR $E26D   E3D1  JSR $E271   sinon,changer le signe
E3D0  LDA #12     E3D4  LDA #16
E3D2  LDY #E4     E3D6  LDY #E4     indexer table des polynomes
E3D4  JMP $E2F9   E3D8  JMP $E2FD   et calculer ACC1*P(ACC1^2)
```

'TAN' (FONCTION)

Principe: TAN(X)=SIN(X)/COS(X). la tangente nécessite donc le calcul d'un cosinus et d'un sinus, c'est donc assez lent et peu précis.

```
E3D7  JSR $DE9B   E3DB  JSR $DEA3   sauver ACC1 --) #C6-#CA
E3DA  LDA #00     E3DE  LDA #00
E3DC  STA 2D      E3E0  STA 2D      signe positif
E3DE  JSR $E38E   E3E2  JSR $E392   calculer SIN
E3E1  LDX #BD     E3E5  LDX #BD
E3E3  LDY #00     E3E7  LDY #00     indexer #00BD
E3E5  JSR $E384   E3E9  JSR $E388   et y sauver SIN(X)
E3E8  LDA #C6     E3EC  LDA #C6
```

```
E3EA  LDY #00     E3EE  LDY #00     indexer #00C6 (X)
E3EC  JSR $DE73   E3F0  JSR $DE7B   récupérer dans ACC1
E3EF  LDA #00     E3F3  LDA #00
E3F1  STA D5      E3F5  STA D5      indiquer signe positif
E3F3  LDA 2D      E3F7  LDA 2D      reprendre signe
E3F5  JSR $E3FF   E3F9  JSR $E403   et calculer COS(X)
E3F8  LDA #BD     E3FC  LDA #BD
E3FA  LDY #00     E3FE  LDY #00     indexer SIN(X)
E3FC  JMP $DDE0   E400  JMP $DDE4   et faire SIN(X)/COS(X)
E3FF  PHA         E403  PHA
E400  JMP $E3C0   E404  JMP $E3C4
```

CONSTANTES POUR SIN

```
E403 E407  BYT #81,#49,#0F,#DA,#A2  soit 1,57079633   ou encore PI/2
E408 E40C  BYT #83,#49,#0F,#DA,#A2  soit 6,28318531   ou encore 2*PI
E40D E411  BYT #7F,#00,#00,#00,#00  soit 0,25         ou encore 1/4
```

DONNEES DU POLYNOME DE CALCUL DE ATN(X)

```
E412 E416  BYT #05
E413 E417  BYT #84,#E6,#1A,#2D,#1B  soit -14,3813907    Coefficient A5
E418 E41C  BYT #86,#28,#07,#FB,#F8  soit  42,0077971    Coefficient A4
E41D E421  BYT #87,#99,#68,#89,#01  soit -76,7041703    Coefficient A3
E422 E426  BYT #87,#23,#35,#DF,#E1  soit  81,6052237    Coefficient A2
E427 E42B  BYT #86,#A5,#5D,#E7,#28  soit -41,3417021    Coefficient A1
E42C E430  BYT #83,#49,#0F,#DA,#A2  soit   6,28318531   Coefficient A0
```

COPYRIGHT MICROSOFT

```
E431 E435  BYT #A1,#54,#46,#8F,#13
E436 E43A  BYT #8F,#52,#43,#89,#CD
```

Ou encore

```
E431 E435  BYT '!'+#80,'T','F','O'+#40
E435 E439  BYT 'S'-#40
E436 E43A  BYT 'O'+#40,'R','C','I'+#40,'M'+#80
```

Et en clair

```
E431 E435  BYT '!TFOSORCIM'
```

'ATN' (FONCTION)

Principe: pour se ramener à un domaine de convergence de Ø à 1, l'égalité classique ARCTG(X)=PI/2-ARCTG(1/X).

Si X est plus grand que 1, on calculera donc l'ARCTG de son inverse qui, lui, sera plus petit que 1 (le tout en valeur absolue bien entendu).

On utilise aussi l'égalité ARCTG(-X)=ARCTG(X) pour traiter les nombres négatifs.

Figure E4-A

La dérivée de Arctg(x) est : $Arctg(x)' = -\frac{1}{1+x^2}$

Un D.L de $\frac{1}{1+x^2}$ est donc : $1 - x^2 + x^4 + \ldots + (-1)^p x^{2p}$

Il suffit d'intégrer ce D.L pour obtenir un D.L de arctg(x) :

$$Arctg(x) = x - \frac{x^3}{3} + \frac{x^5}{5} + \ldots + (-1)^p \frac{x^{2p+1}}{2p+1}$$

Les coéficients de ce développement sont donc voisins de $(-1)^p \frac{1}{2p+1}$

```
E43B  LDA D5      E43F  LDA D5      prendre signe
E43D  PHA         E441  PHA         et sauver
E43E  BPL E443    E442  BPL E447    si positif, c'est bon
E44Ø  JSR $E26D   E444  JSR $E271   sinon, changer le signe
E443  LDA DØ      E447  LDA DØ      prendre exposant
E445  PHA         E449  PHA         sauver aussi
E446  CMP #81     E44A  CMP #81     supérieur à 1 ?
E448  BCC E451    E44C  BCC E455    non, c'est bon
E44A  LDA #4B     E44E  LDA #81
E44C  LDY #DC     E45Ø  LDY #DC     oui, indexer la valeur 1
E44E  JSR $DDEØ   E452  JSR $DDE4   et calculer 1/X
E451  LDA #6B     E455  LDA #6F
E453  LDY #E4     E457  LDY #E4     prendre l'adresse du polynome
E455  JSR $E2F9   E459  JSR $E2FD   et le calculer
E458  PLA         E45C  PLA         récupérer l'exposant
E459  CMP #81     E45D  CMP #81     était-il supérieur à 2 ?
E45B  BCC E464    E45F  BCC E468    non, on a déjà le résultat (valeur absolue)
```

```
E45D  LDA #Ø3      E461  LDA #Ø7
E45F  LDY #E4      E463  LDY #E4      indexer PI/2
E461  JSR $DA8Ø    E465  JSR $DBØB    et calculer ATN(1/X)-PI/2
E464  PLA          E468  PLA          récupérer signe
E465  BPL E46A     E469  BPL E46E     si positif, c'est bon
E467  JMP $E26D    E46B  JMP $E271    sinon, inverser le signe

E46A  RTS          E46E  RTS
```

DONNEES DU POLYNOME DE CALCUL DE ATN

```
E46B E46F  BYT #ØB
E46C E47Ø  BYT #76,#B3,#83,#BD,#D3  soit -Ø,ØØØ684793912  Coéfficient A11
E471 E475  BYT #79,#1E,#F4,#A6,#F5  soit  Ø,ØØ484Ø94216  Coéfficient A1Ø
E476 E47A  BYT #7B,#83,#FC,#BØ,#1Ø  soit -Ø,Ø131117Ø18    Coéfficient A9
E47B E47F  BYT #7C,#ØC,#1F,#67,#CA  soit  Ø,Ø342Ø9638    Coefficient A8
E48Ø E484  BYT #7C,#DE,#53,#CB,#C1  soit -Ø,Ø542791328    Coefficient A7
E485 E489  BYT #7D,#14,#64,#7Ø,#4C  soit  Ø,Ø724971965    Coefficient A6
E48A E48E  BYT #7D,#B7,#EA,#51,#7A  soit -Ø,Ø898Ø23954    Coefficient A5
E48F E493  BYT #7D,#63,#3Ø,#88,#7E  soit  Ø,11Ø932413    Coefficient A4
E494 E498  BYT #7E,#92,#44,#99,#3A  soit -Ø,142839Ø8Ø    Coefficient A3
E499 E49D  BYT #7E,#4C,#CC,#91,#C7  soit  Ø,19999912     Coefficient A2
E49E E4A2  BYT #7F,#AA,#AA,#AA,#13  soit -Ø,333333316    Coefficient A1
E4A3 E4A7  BYT #81,#ØØ,#ØØ,#ØØ,#ØØ  soit  1             Coefficient AØ
```

F) ROUTINES D'ENTREE/SORTIE CASSETTE

1-La fiabilité

Le magnétocassette est la mémoire de masse priviliégée des micro-ordinateurs, de par évidemment son faible cout.

Malheureusement, ce support souffre de deux défauts, plus ou moins importants selon l'utilisation que l'on fait de son ordinateur, à savoir la lenteur et la fiabilité.

Le résultat est d'autant plus fiable que la sauvegarde est lente, c'est pourquoi l'Oric offre deux vitesses de sauvegarde: 3ØØ ou 24ØØ bauds (bit par seconde). Ces vitesses sont en fait des moyennes, ainsi que nous le verrons plus loIn.

Une mauvaise fiabilite est due à plusieurs causes:

-Le support:

La bande magnétique peut présenter des 'Drop out' (irrégularité su revêtement). A ce sujet, il faut noter qu'il ne sert à rien de ralentir la vitesse, car on augmente ainsi la longueur de l'enregistrement, et par là même la probabilité de rencontrer des 'Drops out'.

Une bande a aussi des limites physiques, qui leur interdisent la restitution des fréquences trop élevées.

Cette limitation en fréquence ne joue pas pour les fréquences mises en jeu, du moins au niveau de la bande.

Les cassettes dites 'informatiques' sont en fait des cassettes sélectionnées pour leur absence quasi totale de drop out.

L'enregistreur

Les magnétos généralement utilisés ont une bande passante très limitée (200-5000 Hz environ), ce qui peut nuire à un enregistrement à 2400 bauds.

Leur azimutage (angle entre la tête de lecture/écriture et la bande) varie souvent avec le temps (déréglage), et presque à coup sur d'un magnétophone à l'autre, ce qui se traduit par une réduction très notable de la bande passante.

Si l'enregistreur règle le niveau trop bas, le souffle inhérent à toute bande magnétique peut aussi interférer avec le signal.

L'Oric:

Pour rendre le signal utilisable par le VIA, le signal est filtré et amené à un niveau correct (5V).Cette circuiterie améliore pas mal le signal.

Le logiciel:

Il y a enfin le format utilisé, et l'éventuelle procédure de correction des erreurs. Nous verrons que sur ce point, ORIC n'a pas fait le bon choix...

2-Format d'un octet

Format d'un bit, mode Fast

C'est ici que le format utilisé pèche par son manque d'optimisation: on enregistre à 2400 bauds, avec une fiabilité qui devrait être celle d'un enregistrement à 4800 bauds...

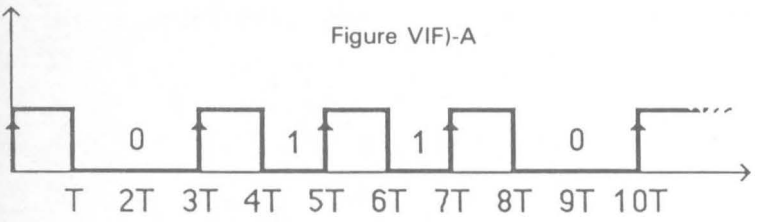
L'écriture est faite par des impulsions successives, dont l'intervalle permet de déterminer la nature du bit reçu.

L'intervalle de 208 microsecondes sera noté T.

Avant chaque donnée, une impulsion de longueur T est envoyée, le bit suivant ensuite, caractérisée par une impulsion de T pour un 1 ou de 2T pour un Ø. (n'oublions pas que chaque impulsion inverse le niveau sur PB7).

Ce qui donne le chronogramme suivant, pour l'émission de Ø11 par exemple.

Figure VIF)-A



Lflèches représentent les fronts détectés par CB1

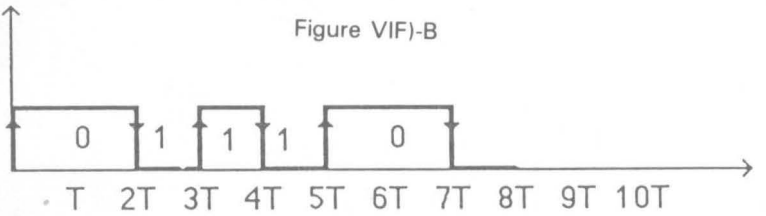
Quelques remarques s'imposent:

-Il faut 2T pour écrire un 1 et 3T pour écrire un Ø. La fréquence oscille donc entre 1600 et 2400 Hz, tout comme le taux de baud réel.

-On peut calculer la vitesse réelle en mode fast, si on considère un octet 'moyen' composé de 4xØ et 4x1, ce qui donne, avec stop, start et parité: 8x1 5xØ. Le taud de baud est donc: 1/(2T*8/13+3T*5/13)=2000 baud environ, soit 154 octets par seconde, ou 1 Ko toutes les 6,5 secondes environ.

-On perd T à chaque bit, ce qui ne sert pas à grand chose, sinon à ajuster les fronts (CB1 est configurée pour détecter les fronts montant). Le chronogramme suivant, doublerait ou presque la vitesse de transfert, sans alterer la fiabilité. Celà entrainerait une complication puisque CB1 devrait détecter tour à tour une transition positive et une transition négative.

Figure VIF)-B



Les flèches représentent les fronts détectés par CB1

Format d'un bit en mode SLOW

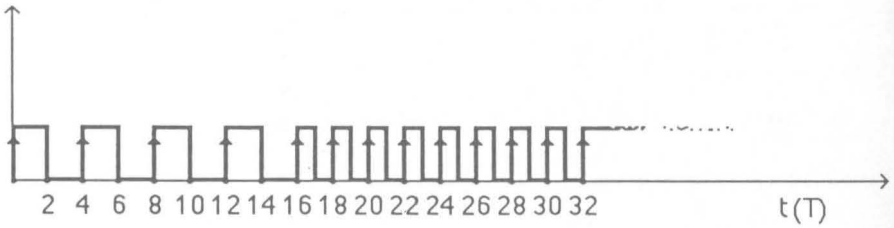
Cette fois, un 1 et un Ø auront globalement la même longueur:
16 crénaux de longueur T pour un 1, ou 8 crénaux de longueur 2T pour un Ø.

Un bit nécessite donc dans tous les cas 16T, soit 3328 microsecondes, ce qui donne 3005 bauds exactement.

La vitesse de transfert réelle est de 23 caractères par seconde, ou encore un Ko toutes les 45 secondes.

Le chronogramme est le suivant, pour la séquence de bit 01 par exemple:

Figure VIF)-C



Le surcroit de fiabilité est en fait apporté par la lecture (voir routines de lecture).

Format d'un octet

Le format est classique, et ressemble à celui utilisé dans toutes les transmissions en série.

Il se compose d'un bit de START, suivi des 8 bits de l'octet, d'un bit de parité inverse, et de 3 bits de stop, ce qui donne:

ST b0 b1 b2 b3 b4 b5 b6 b7 P SP SP SP

Ou encore, pour l'octet #AA par exemple:

0 0 1 0 1 0 1 0 1 1 1 1 1

Il faut donc en tout 13 bits pour un octet...

Il faut considérer que l'octet fait 9 bits, le 9ème, bit de parité inverse, servant de controle. Il est tel que le nonuplet contienne toujours un nombre impair de 1. Si une parité paire avait été choisie, le nonuplet aurait du avoir un nombre de 1 pair.

La parité sert à la lecture pour détecter d'éventuelles erreurs.

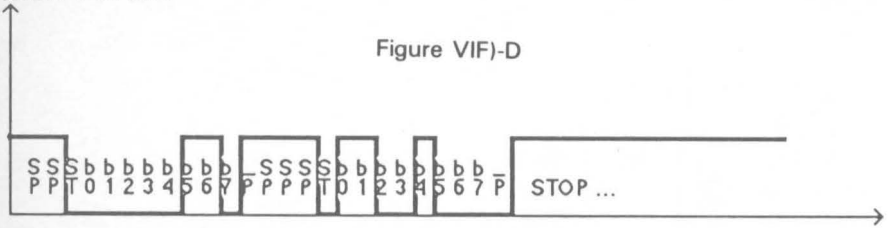
Les bits de stop, au nombre de 3 ici, servent à permettre à l'unité centrale

de traiter l'octet reçu ou à émettre.

Le bit de start, qui est obligatoirement inverse du bit de stop, sert à détecteter le début d'une nouvelle donnée.

Voici le schéma théorique pour l'émission de 2 octets, #60 et #13 par exemple. (Attention,les bits sont représentés par des nivaux logiques, alors que, pour l'oric ils sont représentés par des fréquences différentes ,voir format d'un bit)

Figure VIF)-D



1 1 0 0 0 0 0 0 1 1 0 1 1 1 1 0 1 1 0 0 1 0 0 0 0 1 1

3-Format d'un programme

De même que la structure d'un bit ou d'un octet, il a fallu normaliser la manière de sauvegarder un programme.

Le format est classique: amorce (synchronisation), entête (informations sur le programme), données (programme proprement dit).

La bande amorce permet de se synchroniser, opération qui ne se fait pas nécéssairement du premier coup. Pour marquer le début de l'entête (9 octets), un octet #24 est écrit.

L'entête contient toutes les informations nécéssaires sur le programme, ainsi qu'elles sont décrites ci-dessous.

Suit ensuite le nom, terminé par un Ø. Si le nom n'est pas précisé, il est réduit au Ø.

Le programme est ensuite stocké telquel, sa longueur étant déterminée par l'entête, il n'a pas besoin d'indicateur de fin.

Notons AM l'octet utilisé pour la bande amorce (#16), I l'identificateur d'entête, E0 à E8 les octets de l'entête, Nx les caractères du nom,et Dx les données. Le programme a le format suivant:

259xAM I E0 E1 E2 E3 E4 E5 E6 E7 E8 (N0...Nx) 00 D0 D1 .....

AM = #16
I = #24

E0 = inutilisé
E1-E2 = Adresse de début
E3-E4 = Adresse de fin
E5 = &00=Basic, &80=bloc, &40=tableau
E6 = Drapeau AUTO oui: <>0
E7 = Drapeau Chaine
E8 = Drapeau Entier (uniquement pour V1.1:STORE/RECALL)
E9 = Drapeau FAST oui: <>0

Format d'un tableau

Le format d'un tableau diffère peu de celui d'un programme, sinon que les données sont stockées différemment.

Etant donnée la structure d'un tableau, il est nécéssaire de distinguer les tableaux de chaines et les autres.

-Tableaux réels ou entiers

Ils tiennent en un seul bloc, dans la zone des tableaux. Dans ce cas E1-E2 contient l'adresse du tableau et E3-E4 la fin du tableau.

Mais il se peut très bien que lors du chargement, le tableau ne soit pas à la même adresse dans la zone des tableaux. Le RECALL force le tableau à se charger à la nouvelle adresse, E1-E2 et E3-E4 ne servant en fait qu'à déterminer la longueur du tableau.

-Tableaux de chaines

Ils sont en deux parties: la zone des descripteurs, qui est traitée comme pour les tableaux numériques (y compris relocation au chargement), et les chaines elles-même, qui sont stockées en haut de la mémoire.

Il n'est pas question cette fois de sauver un bloc mémoire, puisque les chaines se trouvent n'importe où, y compris parfois dans le Basic lui même.

Le STORE va donc décrire les descripteurs un à un, et sauver les chaines les une après les autres, sans séparateur.

Le RECALL, après avoir chargé les descripteurs, comme pour un tableau numérique, va les parcourir, réserver la place pour les chaines, et enfin charger le chaines.

SOUS ROUTINE CLOAD

```
E4A8  JSR $E563   ..............  effacer la ligne de status
E4AB  LDA #Ø3     ..............
E4AD  LDY #E5     ..............  indexer 'Searching ..'
E4AF  JSR $E576   ..............  et l'afficher
```

PRENDRE ENTETE DU PROGRAMME

Remarque: La routine est morcelée pour la V1.1

Deux versions circulent pour la V1.1, la version présentée est la version définitive. La première version ne comportait pas le STX #2B1, et des erreurs pendant la bande amorce donnaient donc le message 'error founds'.

La place du STX #2B1 a été gagnée par une simple optimisation, de sorte que les deux V1.1 ne diffèrent que dans la zone #E4B6-#E4D1.

```
E4B2  JSR $E696   E4AC  JSR $E735   attendre la bande amorce
E4B5  JSR $E63Ø   E4AF  JSR $E6C9   prendre un octet
E4B8  CMP #24     E4B2  CMP #24     est-ce #24 ? (indique le début de bloc)
E4BA  BNE E4B5    E4B4  BNE E4AF     non,attendre
..............    E4B6  STX Ø2B1    remettre à Ø indicateur d'erreur
E4BC  LDX #Ø9     E4B9  LDX #Ø9     l'entête fait 9 octets
E4BE  JSR $E63Ø   E4BB  JSR $E6C9   prendre un octet
E4C1  STA 5D,X    E4BE  STA Ø2A7,X  et sauver l'entête en mémoire
E4C3  DEX         E4C1  DEX
E4C4  BNE E4BE    E4C2  BNE E4BB
E4C6  JSR $E63Ø   E4C4  JSR $E6C9   premier caractère du nom
E4C9  BEQ E4DØ    E4C7  BEQ E4D3    fin du nom: on saute
..............    E4C9  CPX #1Ø     a-t-on déjà 16 caractères pour le nom ?
..............    E4CB  BCS E4C4    oui, on charge mais on ne recopie pas
E4CB  STA 49,X    E4CD  STA Ø293,X  sauver le nom
E4CD  INX         E4DØ  INX         indexer caractère suivant
E4CE  BNE E4C6    E4D1  BNE E4C4    inconditionnel: continuer
E4DØ  STA 49,X    E4D3  STA Ø293,X  sauver aussi le terminateur (#ØØ)
..............    E4D6  JSR $E594   afficher 'Found ..'
E4D2  JSR $E6FØ   E4D9  JSR $E79Ø   tester si le nom correspond à celui demandé
E4D5  TXA         E4DC  TXA
E4D6  BNE E4A8    E4DD  BNE E4AC    non, on recommence
..............    E4DF  RTS
E4D8  JSR $E563   ..............  effacer ligne Ø
```

```
E4DB  LDA #12     ..............
E4DD  LDY #E5     ..............  indexer 'Loading  ..'
E4DF  JSR $E576   ..............  et afficher
E4E2  JSR $E56E   ..............  afficher nom du programme demandé
E4E5  NOP         ..............  ?
E4E6  NOP         ..............  ??
E4E7  SBC EA      ..............  ???
E4E9  NOP         ..............  ????
E4EA  NOP         ..............  ?????
```

CHARGER UN PROGRAMME

```
E4EB  LDA 5F      E4E0  LDA 02A9
E4ED  LDY 60      E4E3  LDY 02AA    prendre adresse de début
E4EF  STA 33      E4E6  STA 33
E4F1  STY 34      E4E8  STY 34      et on sauve dans pointeur de travail
E4F3  LDY #00     E4EA  LDY #00     préparer index
E4F5  JSR $E630   E4EC  JSR $E6C9   charger un octet
E4F8  NOP         ..............  ?
E4F9  BCS E53E    ..............  si erreur de format,'FILE ERROR ...
E4FB  STA (33),Y  ..............  sauver l'octet en mémoire
..............    E4EF  LDX 025B    prendre indic charger/vérifier
..............    E4F2  BNE E4F9    vérifier: on saute
..............    E4F4  STA (33),Y  on écrit l'octet en mémoire
..............    E4F6  JMP $E505   et on passe au suivant (BEQ aurait marché)
..............    E4F9  CMP (33),Y  on compare à l'octet en mémoire
..............    E4FB  BEQ E505    si OK, au suivant
..............    E4FD  INC 025C    sinon, on incrémente le compteur d'erreurs
..............    E500  BNE E505
..............    E502  INC 025D    poids fort aussi
E4FD  JSR $E554   E505  JSR $E56C   octet suivant et test de fin
E500  BCC E4F5    E508  BCC E4EC    pas fini, on recommence
E502  RTS         E50A  RTS
```

TABLE DES MESSAGES POUR CHARGEMENT

```
E503 E50B  BYT #10,#07
E505 E50D  BYT "Searching .."
E511 E519  BYT #00
E512 E51A  BYT #10,#07
E514 E51C  BYT "Loading  .."
E520 E526  BYT #00
E521 ....  BYT #0A,#0D
```

```
E523 ....  BYT 'FILE ERROR / LOAD'
E534 ....  BYT ' ABORTED'
E53C ....  BYT #ØD,#ØØ
.... E527  BYT #ØA,#ØD
.... E529  BYT 'Errors found'
.... E535  BYT #ØD,#ØA,#ØØ
.... E538  BYT #1Ø,#Ø7
.... E53A  BYT 'Found ..'
.... E542  BYT #ØØ
.... E543  BYT #1Ø,#Ø7
.... E545  BYT 'Verifying ..'
.... E551  BYT #ØØ
.... E552  BYT ' Verify errors'
.... E56Ø  BYT ' detected'
.... E569  BYT #ØD,#ØA,#ØØ
```

AFFICHER 'FILE ERROR ...

```
E53E  JSR $E563  ..............  effacer ligne Ø
E541  JSR $C719  ..............  faire 'NEW'
E544  JSR $E8Ø4  ..............  reconfigurer le VIA
E547  JSR $CB9F  ..............  aller à la ligne
E54A  LDA #21    ..............
E54C  LDY #E5    ..............  indexer le message
E54E  JSR $CBED  ..............  l'afficher
E551  JMP $C4B5  ..............  et directement à l'interpréteur
```

PASSER OCTET SUIVANT,TEST SI FIN

Entrée: #33-#34 contient l'adresse de l'octet courant, et #61-#62/#2AB-#2AC contient le dernier octet du programme

Sortie: C=1 si le dernier octet est dépassé par l'incrémentation, C=Ø sinon
#33-#34 est incrémenté
X et Y sont conservés

```
E554  LDA 33      E56C  LDA 33      prendre adresse courante
E556  CMP 61      E56E  CMP Ø2AB    et faire pseudo différence
E558  LDA 34      E571  LDA 34      avec l'adresse de fin
E55A  SBC 62      E573  SBC Ø2AC    C est placé comme pour la différence
E55C  INC 33      E576  INC 33      incrémenter de toutes faÇons le pointeur
E55E  BNE E562    E578  BNE E57C    Rappel:INC ne touche pas à C ...
```

```
E56Ø  INC 34      E57A  INC 34
E562  RTS         E57C  RTS


                        AFFICHER 'Searching ..'

.............. E57D  LDA #ØB
.............. E57F  LDY #E5     indexer 'Searching ..'
.............. E581  JSR $E5EA   et afficher
.............. E584  RTS


                  AFFICHER 'Saving ..' ET LE NOM DU PROGRAMME

.............. E585  LDA #45
.............. E587  LDY #E6     indexer 'Saving ..'
.............. E589  JSR $E5EA   et afficher
.............. E58C  LDA #7F
.............. E58E  LDY #Ø2     indexer le nom (#Ø27F)
.............. E59Ø  JSR $E5B6   et l'afficher
.............. E593  RTS

                  AFFICHER 'Found ..' ET LE NOM DU PROGRAMME

.............. E594  LDA #38
.............. E596  LDY #E5     indexer 'Found ..'
.............. E598  JMP $E5AB   miracle:Pas JSR/RTS !.BNE aurait été mieux

          AFFICHER 'Loading' OU 'Verifying ..' ET LE NOM DU PROGRAMME

.............. E59B  LDA Ø25B    prendre indicateur de mode
.............. E59E  BNE E5A7    sauter si vérification
.............. E5AØ  LDA #1A
.............. E5A2  LDY #E5     indexer 'Loading ..'
.............. E5A4  JMP $E5AB   ...ou BNE:afficher
.............. E5A7  LDA #43
.............. E5A9  LDY #E5     indexer 'Verifying ..'
.............. E5AB  JSR $E5EA   afficher le message
.............. E5AE  LDA #93
.............. E5BØ  LDY #Ø2     indexer nom du programme trouvé (#Ø293)
.............. E5B2  JSR $E5B6   et l'afficher
.............. E5B5  RTS
```

AFFICHER NOM DU PROGRAMME ET SON TYPE

Entrée: AY contient l'adresse du nom

Sortie: Rien de spécial...

```
..............  E5B6  JSR $F865   afficher le nom indexé par AY
..............  E5B9  INX         se placer sur la colonne suivante
..............  E5BA  LDY #00     indexer B(asic)
..............  E5BC  STY 025F    placer terminateur du type (#00)
..............  E5BF  LDA 02AE    prendre drapeau BASIC
..............  E5C2  BEQ E5D7    oui,c'est bon
..............  E5C4  INY         indexer C(ode)
..............  E5C5  BIT 02AE    tester pour machine
..............  E5C8  BMI E5D7    oui,c'est bon
..............  E5CA  INY         indexer S(tring)
..............  E5CB  BIT 02AF    et tester drapeau chaine
..............  E5CE  BMI E5D7    oui,c'est bon
..............  E5D0  INY         indexer I(nteger)
..............  E5D1  BIT 02B0    tester drapeau entier
..............  E5D4  BMI E5D7    oui,c'est bon
..............  E5D6  INY         non,c'est R(eal)
..............  E5D7  LDA E5E5,Y  prendre indicateur
..............  E5DA  STA 025E    et le sauver à l'adresse
..............  E5DD  LDA #5E
..............  E5DF  LDY #02     indexer adresse du type (#025E)
..............  E5E1  JSR $F865   et afficher le type
..............  E5E4  RTS

....  E5E5  BYT "B","C","S","I","R"
```

ECRIRE AU DEBUT DE LA LIGNE 0

Entrée: AY indexe le message à afficher (il doit être terminé par un NULL (#00)

Sortie: le message est affiché (sauf en mode HIRES) et X pointe 2 colonnes plus loin que la fin du message.

```
..............  E5EA  JSR $E5F5   effacer la ligne 0
..............  E5ED  LDX #00     indiquer commencer colonne 0
..............  E5EF  JSR $F865   et afficher le message
..............  E5F2  INX
..............  E5F3  INX         se placer 2 colonne plus loin
..............  E5F4  RTS
```

EFFACER LA LIGNE Ø

Entrée: rien de spécial

Sortie: A conservé sur la V1.1 (?)

Bogue: la ligne est effacée même en mode HIRES sur la V1.Ø

```
.............. E5F5  PHA         sauver A
.............. E5F6  LDA Ø21F     prendre drapeau HIRES
.............. E5F9  BNE E6Ø5     oui,on sort
E563  LDX #1C    E5FB  LDX #22     effacer 28/34 caractères
E565  LDA #1Ø    E5FD  LDA #1Ø     prendre attribut fond noir
E567  STA BB8Ø,X E5FF  STA BB8Ø,X  et effacer
E56A  DEX        E6Ø2  DEX
E56B  BPL E567   E6Ø3  BPL E5FF
.............. E6Ø5  PLA         récupérer A
E56D  RTS        E6Ø6  RTS
```

AFFICHER NOM DU PROGRAMME

```
E56E  INX        ..............
E56F  INX        ..............
E57Ø  INX        .............. se placer 3 colonnes après le message
E571  LDA #35    ..............
E573  LDY #ØØ    .............. indexer le nom demandé (#ØØ35)
E575  BYT #2C    .............. et sauter l'instruction suivante
```

AFFICHER AU DEBUT DE LA LIGNE Ø

Entrée: AY indexe la message à afficher

Sortie: X pointe sur la 1ère colonne libre après le message.

```
E576  LDX #ØØ    .............. se placer au début de la ligne (colonne Ø)
E578  JMP $F436  .............. et afficher le message
```

SOUS ROUTINE CSAVE/SAUVER L'ENTETE

```
E57B  JSR $E6BA  E6Ø7  JSR $E75A  sauver la bande amorce
E57E  LDA #24    E6ØA  LDA #24
```

```
E58Ø  JSR $E5C6   E6ØC  JSR $E65E   et l'indicateur de début d'entête
E583  LDX #Ø9     E6ØF  LDX #Ø9
E585  LDA 5D,X    E611  LDA Ø2A7,X  prendre l'entête
E587  JSR $E5C6   E614  JSR $E65E   et sauver sur la cassette
E58A  DEX         E617  DEX
E58B  BNE E585    E618  BNE E611    sortie X=Ø
E58D  LDA 35,X    E61A  LDA Ø27F,X  prendre le nom
E58F  BEQ E597    E61D  BEQ E625    si dernier caractère, on saute
E591  JSR $E5C6   E61F  JSR $E65E   sauver le nom
E594  INX         E622  INX         indexer caractère suivant
E595  BNE E58D    E623  BNE E61A    inconditionnel: on continue
E597  JSR $E5C6   E625  JSR $E65E   sauver aussi le terminateur (#ØØ)
E59A  JSR $E563   ..............    effacer la ligne de status
E59D  LDA #BC     ..............
E59F  LDY #E5     ..............    indexer le message 'Save ..'
E5A1  JSR $E576   ..............    et l'afficher
E5A4  JSR $E56E   ..............    afficher le nom du programme
..............    E628  LDX #ØØ
..............    E62A  DEX         attendre environ 128Ø µs
..............    E62B  BNE E62A    (le temps d'affichage du nom à la lecture)
..............    E62D  RTS
```

SAUVER UN PROGRAMME SUR CASSETTE

```
E5A7  LDA 5F      E62E  LDA Ø2A9
E5A9  LDY 6Ø      E631  LDY Ø2AA    prendre adresse de début
E5AB  STA 33      E634  STA 33
E5AD  STY 34      E636  STY 34      et sauver dans pointeur de travail
E5AF  LDY #ØØ     E638  LDY #ØØ     index =Ø
E5B1  LDA (33),Y  E63A  LDA (33),Y  prendre octet dans la mémoire
E5B3  JSR $E5C6   E63C  JSR $E65E   et sauver sur cassette
E5B6  JSR $E554   E63F  JSR $E56C   incrémenter et test si fin
E5B9  BCC E5B1    E642  BCC E63A    non, on continue
E5BB  RTS         E644  RTS
```

MESSAGE POUR SAUVEGARDE

```
E5BC E645  BYT #1Ø,#Ø7
E5BE E647  BYT 'Saving '
.... E64E  BYT '..'
E5C5 E65Ø  BYT #ØØ
```

TESTER ERREUR DE FORMAT

```
.....  E651  LDA Ø2B1    tester drapeau d'erreur
.....  E654  BEQ E65D    si Ø, c'est bon
.....  E656  LDA #27
.....  E658  LDY #E5     indexer 'error found ...'
.....  E65A  JSR $CCBØ   et afficher le message
.....  E65D  RTS
.....  E65E  STA 2F
```

ECRIRE UN OCTET SUR LA K7

Entrée: A contient l'octet à envoyer

Sortie: l'octet est envoyé après synchro, sous le format:
Ø bØ b1 b2 b3 b4 b5 b6 b7 P 1 1 1 (P représente la parité inverse)

```
E5C6  STA 2F      E65E  STA 2F      sauver l'octet à envoyer
E5C8  TXA         E66Ø  TXA
E5C9  PHA         E661  PHA         sauver X
E5CA  TYA         E662  TYA
E5CB  PHA         E663  PHA         sauver Y
E5CC  JSR $E627   E664  JSR $E6CØ  synchro: attendre créneau précedent terminé
E5CF  CLC         E667  CLC         bit de start=Ø
E5DØ  LDY #Ø9     E668  LDY #Ø9     indiquer 9 octets
E5D2  LDA #ØØ     E66A  LDA #ØØ     initialiser la parité
E5D4  BEQ E5DC    E66C  BEQ E674    inconditionnel: écrire bit de start (Ø)

E5D6  LSR 2F      E66E  LSR 2F      sortir bØ,b1... dans C
E5D8  PHP         E67Ø  PHP         sauver C
E5D9  ADC #ØØ     E671  ADC #ØØ     calculer la parité
E5DB  PLP         E673  PLP         récupérer le bit
E5DC  JSR $E5F3   E674  JSR $E68B   et envoyer bit dans C
E5DF  DEY         E677  DEY         octet envoyé ?
E5EØ  BNE E5D6    E678  BNE E66E    non, on continue
E5E2  EOR #Ø1     E67A  EOR #Ø1     oui, on inverse la parité
E5E4  LSR A       E67C  LSR A       et on la sort dans C
E5E5  LDY #Ø4     E67D  LDY #Ø4     1 parité+3 bits de stop (1)
E5E7  JSR $E5F3   E67F  JSR $E68B   envoyer bit
E5EA  SEC         E682  SEC         bit stop=1
E5EB  DEY         E683  DEY         bits de stop envoyés ?
E5EC  BNE E5E7    E684  BNE E67F    non, on continue
```

```
E5EE  PLA          E686  PLA
E5EF  TAY          E687  TAY          récupérer Y
E5F0  PLA          E688  PLA
E5F1  TAX          E689  TAX          récupérer X
E5F2  RTS          E68A  RTS
```

ECRIRE UN BIT SUR LA K7

Entrée: C contient le bit à envoyer

Sortie: A,Y inchangés.

```
E5F3  PHA          E68B  PHA          sauver A
E5F4  PHP          E68C  PHP          et C (qui contient la valeur du bit)
E5F5  LDA 67       E68D  LDA Ø24D      tester mode SLOW ou FAST
E5F7  BNE E603     E690  BNE E69C     et sauter si SLOW
E5F9  SEC          E692  SEC          FAST: indiquer longueur créneau=208 µs
E5FA  JSR $E619    E693  JSR $E6B2    et envoyer premier créneau
E5FD  PLP          E696  PLP          récupérerle bit dans C
E5FE  JSR $E619    E697  JSR $E6B2    envoyer créneau de 416 µs (Ø) 208 µs (1)
E601  PLA          E69A  PLA          récupérer A
E602  RTS          E69B  RTS

E603  JSR $E619    E69C  JSR $E6B2    SLOW: envoyer créneau, longueur dans C
E606  LDX #0F      E69F  LDX #0F      préparer pour 16 créneaux
E608  PLP          E6A1  PLP          récupére le bit
E609  BCS E60D     E6A2  BCS E6A6
E60B  LDX #07      E6A4  LDX #07      si Ø, 8 créneaux seulement
E60D  JSR $E612    E6A6  JSR $E6AB    et envoyer X créneaux
E610  PLA          E6A9  PLA          récupérer A
E611  RTS          E6AA  RTS
```

Ecrire X créneaux, dont la longueur est dans le latche de T1.

```
E612  JSR $E627    E6AB  JSR $E6C0    synchro, et relancer le décompte
E615  DEX          E6AE  DEX          fini ?
E616  BNE E612     E6AF  BNE E6AB     non, on recommence
E618  RTS          E6B1  RTS
```

ECRIRE UN CRENEAU SUR LA K7

Entrée: C contient la longueur du créneau

Sortie: Y inchangé.

```
E619  LDA #$D0     E6B2  LDA #$D0
E61B  LDX #$00     E6B4  LDX #$00     AX=208 µs, longueur pour 1
E61D  BCS E621     E6B6  BCS E6BA     si c'est 1, on saute
E61F  ASL A        E6B8  ASL A
E620  INX          E6B9  INX          sinon, AX=416 µs, longueur pour 0
E621  STA $0306    E6BA  STA $0306    écrire dans le latche
E624  STX $0307    E6BD  STX $0307    et démarrer le comptage, met à 0 IFR
```

Synchronisation

```
E627  LDA $0304    E6C0  LDA $0304    inutlile

E62A  BIT $030D    E6C3  BIT $030D    tester b6, indicateur T1=0
E62D  BVC E62A     E6C6  BVC E6C3     si pas encore 0, on repart
E62F  RTS          E6C8  RTS
```

PRENDRE UN OCTET SUR LA K7

Entrée: rien de spécial...

Sortie: A contient l'octet (Z et N sont positionnés en conséquense), X et Y inchangés.

V1.0: les erreurs de parité sont curieusement ignorées, bien que la parité soit calculée !. En mode SLOW, une erreur de synchro mettra C=1, C=0 sinon.

V1.1: si erreur de synchro (SLOW) ou de parité, b7 de $2B1 est forcé à 1.

```
E630  TYA          E6C9  TYA
E631  PHA          E6CA  PHA          sauver Y
E632  TXA          E6CB  TXA
E633  PHA          E6CC  PHA          sauver X
E634  JSR $E67D    E6CD  JSR $E71C    se synchroniser sur le premier créneau
E637  JSR $E67D    E6D0  JSR $E71C    prendre longueur d'un créneau
E63A  BCS E637     E6D3  BCS E6D0     c'est 1, c'est un bit de stop, on attend
E63C  JSR $E661    E6D5  JSR $E6FF    synchro pour SLOW
E63F  BCS E657     E6D8  BCS E6F0     sortir si erreur de format
E641  LDA #$00     E6DA  LDA #$00     parité=0
E643  LDY #$08     E6DC  LDY #$08     indiquer 8 bits
E645  JSR $E65E    E6DE  JSR $E6FC    prendre un bit dans C
```

```
E648  PHP          E6E1  PHP          sauver sa valeur
E649  ROR 2F       E6E2  ROR 2F       assembler l'octet
E64B  PLP          E6E4  PLP          récupérer bit
E64C  ADC #00      E6E5  ADC #00      calculer la parité
E64E  DEY          E6E7  DEY          et continuer jusqu'à la fin de l'octet
E64F  BNE E645     E6E8  BNE E6DE
E651  JSR $E65E    E6EA  JSR $E6FC    prendre bit de parité
E654  ADC #00      ..................  et ajouter à parité
E656  CLC          ..................  indiquer résultat OK, quoiqu'il arrive
..................  E6ED  SBC #00      comparer les parité calculée et reçue
..................  E6EF  LSR A        dans C
..................  E6F0  BCC E6F5     pas d'erreur, on sort
..................  E6F2  ROL 02B1     sinon, b7 =1 (C=1)
E657  PLA          E6F5  PLA
E658  TAX          E6F6  TAX          récupérer X
E659  PLA          E6F7  PLA
E65A  TAY          E6F8  TAY          récupérer Y
E65B  LDA 2F       E6F9  LDA 2F       et octet dans A
E65D  RTS          E6FB  RTS
```

PRENDRE UN BIT SUR LA K7

Entrée: rien

Sortie:C contient la valeur du bit.

Principe: on prend la longueur du créneau courant, et on positionne C en conséquense pour le mode FAST.

Pour le mode SLOW, on va attendre toutes les transitions écrites, leur nombre dépendant de la valeur du créneau. Certains créneaux peuvent être altérés, à condition que le premier (celui du FAST) soit bien lu.

Si le premier créneau est bien lu, les suivants peuvent être lus n'importe comment si c'est un 0, ou si c'est un 1, il suffit que 4 sur 6 soient bien lus.

Si le premier créneau est mal lu on peut retrouver le bon résultat, mais on va se désynchroniser, la suite sera donc problématique...

```
E65E  JSR $E67D    E6FC  JSR $E71C    prendre la longueur du créneau courant
..................  E6FF  PHA          sauver A
E661  LDA 67       E700  LDA 024D     tester FAST/SLOW
E663  BEQ E67C     E703  BEQ E71A     et c'est tout si FAST
```

```
E665  PHA          ..............  sauver A
E666  JSR $E67D    E705  JSR $E71C  SLOW: prendre transition
E669  LDX #02      E708  LDX #02    préparer pour 2 transitions supplémentaires
E66B  BCC E66F     E70A  BCC E70E
E66D  LDX #06      E70C  LDX #06    préparer pour 6 transitions supplémentaires
E66F  LDA #00      E70E  LDA #00    initialiser compteur
E671  JSR $E67D    E710  JSR $E71C  prendre un bit dans C
E674  ADC #00      E713  ADC #00    ajouter á compteur
E676  DEX          E715  DEX        et tester si fin
E677  BNE E671     E716  BNE E710   non, on recommence
E679  CMP #04      E718  CMP #04    positionner C selon résultat
E67B  PLA          E71A  PLA        et récupérer A
E67C  RTS          E71B  RTS
```

PRENDRE LA LONGUEUR D'UN CRENEAU

Entrée: rien

Sortie: C=0 si créneau plus court que 256 µs (1), C=1 si créneau plus long que 512 µs (0). Tolérance de +- 20 % entre la vitesse de lecture et d'écriture donc.

(on peut donc augmenter la vitesse de défilement à la lecture)
A n'est pas touché.

```
E67D  PHA          E71C  PHA          sauver A
E67E  LDA 0300     E71D  LDA 0300     annuler indicateur d'IRQ sur CB1
E681  LDA 030D     E720  LDA 030D     prendre IFR
E684  AND #10      E723  AND #10      et attendre transition reçue
E686  BEQ E681     E725  BEQ E720     non, on attend...
E688  LDA 0309     E727  LDA 0309     prendre poids fort de la longueur
E68B  PHA          E72A  PHA          et on sauve
E68C  LDA #FF      E72B  LDA #FF      et on réinitialise le timer
E68E  STA 0309     E72D  STA 0309     et on relance le décompte
E691  PLA          E730  PLA          récupérer longueur
E692  CMP #FE      E731  CMP #FE      placer C selon résultat
E694  PLA          E733  PLA          et récupérer A
E695  RTS          E734  RTS
```

TROUVER LA BANDE AMORCE

```
E696  JSR $E65E    E735  JSR $E6FC  prendre un bit
E699  ROR 2F       E738  ROR 2F     et assembler l'octet
```

```
E69B  LDA #16     E73A  LDA #16     est-ce #16 (%00010110)
E69D  CMP 2F      E73C  CMP 2F
E69F  BNE E696    E73E  BNE E735    non, on continue la recherche
E6A1  LDA 67      E740  LDA 024D    oui, mode FAST ?
E6A3  BEQ E6AD    E743  BEQ E74D     oui, sauter
E6A5  JSR $E67D   E745  JSR $E71C     non, sauter éventuelle parité
E6A8  JSR $E67D   E748  JSR $E71C   sauter les bits de stop
E6AB  BCS E6A8    E74B  BCS E748
E6AD  LDX #03     E74D  LDX #03     vérifier au moins 3 octets à #16
E6AF  JSR $E630   E74F  JSR $E6C9   prendre un octet
E6B2  CMP #16     E752  CMP #16     c'est #16 ?
E6B4  BNE E696    E754  BNE E735    non, on recommence la synchro
E6B6  DEX         E756  DEX
E6B7  BNE E6AF    E757  BNE E74F    et on fait 3 octets
E6B9  RTS         E759  RTS
```

ECRIRE LA BANDE AMORCE SUR LA K7

Entrée: rien

Sortie: X=0,Y=0,Z=1 et 259 octets (curieux nombre !) sont écrits.

```
E6BA  LDX #02     E75A  LDX #02
E6BC  LDY #03     E75C  LDY #03
E6BE  LDA #16     E75E  LDA #16     prendre %00010110
E6C0  JSR $E5C6   E760  JSR $E65E   et écrire
E6C3  DEY         E763  DEY         boucler sur Y
E6C4  BNE E6BE    E764  BNE E75E
E6C6  DEX         E766  DEX         et sur X
E6C7  BNE E6BE    E767  BNE E75E    soit 3+256=259 octets
E6C9  RTS         E769  RTS
```

CONFIGURER LE VIA POUR TRAVAIL K7

Bogue: génère un STROBE pour l'imprimante

```
E6CA  JSR $EBFD   E76A  JSR $EE1A   interdire IRQ par T1
E6CD  LDY #06     E76D  LDY #06     pour les 7 registres
E6CF  SEI         E76F  SEI         interdire IRQ
E6D0  LDX E6E2,Y  E770  LDX E782,Y  prendre index du registre
E6D3  LDA E6E9,Y  E773  LDA E789,Y  et valeur à y placer
E6D6  STA 0300,X  E776  STA 0300,X  écrire la valeur dans le registre
```

```
E6D9  DEY          E779  DEY          et suivant...
E6DA  BPL E6D0     E77A  BPL E770
E6DC  LDA #40      E77C  LDA #40      fermer le relais de télécommande
E6DE  STA 0300     E77E  STA 0300     (et générer un strobe !)
E6E1  RTS          E781  RTS
```

DONNEES POUR CONFIGURATION

```
E6E2 E782  BYT #05,#04,#0B,#02,#0C,#08,#0E
E6E9 E789  BYT #00,#D0,#C0,#FF,#10,#F4,#7F
#0305=#00          T1 (timer d'écriture)=#00D0
#0304=#D0          c'est à dire la fréquence d'écriture d'un 1
#030B=#C0          T1 en mode monostable, sortie sur PB7
#0302=#FF          relais en entrée, donc forcé à 1 et donc fermé
#030C=#10          détecter un front montant sur CB1 (entrée K7)
#0308=#F4          ?
#030E=#7F          ne pas déclencher d'IRQ : DO NOT DISTURB !
```

COMPARER NOM DEMANDE ET NOM TROUVE

Entrée: V1.0: le nom à chercher est en #35..., et celui trouvé est en #49..., tous deux terminés par un 0.

V1.1: le nom à chercher est en #27F..., et celui trouvé est en #293..., tous deux terminés par un 0.

Sortie: X=0 si les noms sont les mêmes, ou si le nom demandé était vide.

X<>0 si les noms ne sont pas pareils, le nom trouvé est devenu pareil que le nom à trouver.

Remarque: la comparaison se fait au maximum sur les 16 premiers caractères du nom.

```
E6F0  LDY #00      E790  LDY #00      initialiser index noms
E6F2  LDX #00      E792  LDX #00      et indiquer pareils pour l'instant
E6F4  LDA 35       E794  LDA 027F     si pas de nom demandé
E6F6  BEQ E70D     E797  BEQ E7AE     on sort X=0
E6F8  LDA 0035,Y   E799  LDA 027F,Y   prendre nom demandé
E6FB  CMP 0049,Y   E79C  CMP 0293,Y   et comparer à nom trouvé
E6FE  BEQ E701     E79F  BEQ E7A2     si pareil, suivant
E700  INX          E7A1  INX          si pas pareil, indiquer erreur
E701  STA 0049,Y   E7A2  STA 0293,Y   remplacer nom trouvé par nom désiré
E704  INY          E7A5  INY          caractère suivant
```

```
E7Ø5  CPY #11      E7A6  CPY #11      16 caractères suffisent !
E7Ø7  BCS E7ØD     E7A8  BCS E7AE     et on sort si 16 (+#ØØ) caractères testés.
E7Ø9  PHA          E7AA  PHA          sinon, c'était un Ø ? (terminateur)
E7ØA  PLA          E7AB  PLA          astuce pour positionner Z selon A
E7ØB  BNE E6F8     E7AC  BNE E799     non, on continue à comparer
E7ØD  RTS          E7AE  RTS
```

AFFICHER MESSAGE COPYRIGHT

Remarque: la routine n'est jamais appelée par la ROM, et ANDY BROWN se croit meilleur que PETER HALFORD puisqu'il n'efface pas son nom !

```
E7ØE  JSR $E563    ..............  effacer ligne Ø
E711  LDA #8D      ..............
E713  LDY #EB      ..............  indexer 'Software by...
E715  LDX #ØØ      ..............  à placer à la première colonne
E717  JSR $E576    ..............  afficher le message sur la ligne de status
E71A  JSR $E9Ø5    ..............  attendre la pression d'une touche
E71D  BPL E71A     ..............
E71F  JMP $E563    ..............  et effacer la ligne Ø, sauf ANDY BROWN.
```

SYNTAXE CLOAD ET CSAVE

```
E725  LDA #ØØ      E7B2  LDA #ØØ      initialiser divers indicateurs
E727  STA 67       E7B4  STA Ø24D     mode FAST
E729  STA 63       E7B7  STA Ø2AD     BASIC
E72B  STA 64       E7BA  STA Ø2AE     non AUTO
..............     E7BD  STA Ø25B     pas de vérification
..............     E7CØ  STA Ø25A     pas de merge
..............     E7C3  STA Ø25C
..............     E7C6  STA Ø25D     nombre d'erreur à Ø
..............     E7C9  STA Ø2B1     et indiquer pas d'erreur de format
E72D  JSR $CE8B    E7CC  JSR $CF17    évaluer l'expression
E73Ø  BIT 28       E7CF  BIT 28       tester drapeau chaine
E732  BPL E722     E7D1  BPL E7AF     non, erreur (#D712/#D7CD est là pour Ça)
E734  JSR $D715    E7D3  JSR $D7DØ    prendre descripteur
E737  TAX          E7D6  TAX          longueur dans X
E738  LDY #ØØ      E7D7  LDY #ØØ      préparer index
E73A  INX          E7D9  INX          ajuster longueur
E73B  DEX          E7DA  DEX          longueur-1
E73C  BEQ E746     E7DB  BEQ E7E7     c'est fini, on sort
```

```
E73E  LDA (91),Y  E7DD  LDA (91),Y  prendre caractère dans la chaine
E740  STA 0035,Y  E7DF  STA 027F,Y  et sauver en mémoire
E743  INY         E7E2  INY         préparer caractère suivant
.....             E7E3  CPY #10     si plus de 16 caracteres, ça  suffit
E744  BNE E73B    E7E5  BNE E7DA    continuer
E746  LDA #00     E7E7  LDA #00     prendre NULL (#00) comme terminateur
E748  STA 0035,Y  E7E9  STA 027F,Y  et sauver
E74B  JSR $00E8   E7EC  JSR $00E8   prendre caractère courant
E74E  BEQ E79C    E7EF  BEQ E852    si lus de paramètre, on sort
E750  CMP #','    E7F1  CMP #','    sinon,on veut une virgule
E752  BNE E722    E7F3  BNE E7AF    (#CFD9/D065 n'existait sans doute pas)
E754  JSR $00E2   E7F5  JSR $00E2   sauter la virgule
E757  BEQ E79C    E7F8  BEQ E852    sortir si plus de paramètres
E759  CMP #','    E7FA  CMP #','    si que des virgules,
E75B  BEQ E754    E7FC  BEQ E7F5    ignorer simplement (?)
E75D  LDY #05     .....             préparer compteur de paramètres
E75F  DEY         .....             paramètre précédent
E760  BEQ E722    .....             si pas reconnu, erreur
E762  CMP E7A5,Y  .....             comparer à la table des paramètres
E765  BNE E75F    .....             pas bon, essayer le suivant
E767  CPY #04     .....             est-ce AUTO ?
E769  BCC E76F    .....             non, sauter
E76B  STY 63      .....             oui, #63<>0
E76D  BCS E754    .....             inconditionnel: continuer évaluation
E76F  CPY #03     .....             est-ce S ?
E771  BCC E777    .....             non, sauter
E773  STY 67      .....             oui, #67<>0
E775  BCS E754    .....             inconditionnel
E777  JSR $00E2   .....             c'est E ou A: on saute le caractère
E77A  LDX #80     .....             on indique bloc machine
E77C  STX 64      .....             DEC #64 aurait suffit !
E77E  CPY #02     .....             est-ce A ?
E780  BCC E78F    .....             non, traiter E
E782  JSR $E79D   .....             oui, évaluer un entier
E785  LDA 33      .....             inutile, il est dans YA
E787  LDY 34      .....
E789  STA 5F      .....             et sauver comme adresse de début
E78B  STY 60      .....
E78D  BCC E74B    .....             inconditionnel: continuer évaluation
E78F  JSR $E79D   .....             évaluer un entier
E792  LDA 33      .....
E794  LDY 34      .....             inutile...
E796  STA 61      .....
E798  STY 62      .....             et sauver comme adresse de fin
```

```
E79A  BCC E74B     ..............  inconditionnel
E79C  RTS          ..............
..............  E7FE  CMP #%AUTO  est-ce AUTO ?
..............  E800  BNE E807    non, sauter
..............  E802  STA 02AD    oui, #2AD <>0
..............  E805  BCS E7F5    inconditionnel: continuer
..............  E807  CMP #'S'    SLOW demandé ?
..............  E809  BNE E810    non, sauter
..............  E80B  STA 024D    oui, #24D <> 0
..............  E80E  BCS E7F5    inconditionnel: continuer
..............  E810  CMP #'V'    Vérification demandée ?
..............  E812  BNE E819    non, sauter
..............  E814  STA 025B    oui, #25B <> 0
..............  E817  BCS E7F5    inconditionnel
..............  E819  CMP #'J'    merge (Join) demandé ?
..............  E81B  BNE E822    non, sauter
..............  E81D  STA 025A    oui, #25A <> 0
..............  E820  BCS E7F5    inconditionnel
..............  E822  CMP #'A'    début de bloc ?
..............  E824  BEQ E82A    oui, sauter
..............  E826  CMP #'E'    fin de bloc ?
..............  E828  BNE E871    non 'SYNTAX ERROR'
..............  E82A  STA 0E      sauver code début/fin
..............  E82C  JSR $00E2   sauter le A ou le E
..............  E82F  LDX #80     indiquer pas Basic
..............  E831  STX 02AE
..............  E834  JSR $E853   évaluer un entier
..............  E837  LDA 33      inutile, il est dans YA
..............  E839  LDY 34
..............  E83B  LDX 0E      reprendre l'appelant
..............  E83D  CPX #'A'    c'était A ?
..............  E83F  BNE E849    non, sauter
..............  E841  STA 02A9
..............  E844  STY 02AA    oui, sauver adresse de début
..............  E847  BCS E7EC    inconditionnel
..............  E849  STA 02AB
..............  E84C  STY 02AC    sauver adresse de fin
..............  E84F  JMP $E7EC   et continuer...
..............  E852  RTS
```

EVALUER UN NOMBRE SUR 2 OCTETS

Entrée: TXTPTR correctement placé

Sortie: YA, $33-$34, $D4-$D3 contiennent le nombre (non signé)
C=0,N et Z positionnés selon le poids faible.

```
E79D  JSR $CE77   E853  JSR $CF03   évaluer un nombre dans ACC1
E7A0  JSR $D867   E856  JSR $D922   ACC1 --> Entier non signé
E7A3  CLC         E859  CLC        C=0 (faciliter branchements relatifs ?)
E7A4  RTS         E85A  RTS
```

TABLE DES PARAMETRES PERMIS

Remarque: le '!' est là on se demande pourquoi, l'index ne tombant jamais à 0. Bizarre, bizarre....

```
E7A5 ....  BYT '!','E','A','S',$C7 (AUTO)
```

'CLOAD' (COMMANDE)

Bogues: nombreuses et variées...

-Que le programme soit Basic ou non, le pointeur de fin du Basic est placé à la fin du programme chargé.

-Du fait qu'on utilise le tampon clavier pour sauver les variables systèmes, les retours devaient générer des SYNTAX ERROR assez génant. Pour les éviter, la solution radicale a été choisie: retour direct à l'interpréteur, ce qui interdit l'utilisation en mode programme d'un programme non enregistré en AUTO...

-Les erreurs de parité sont ignorées (voir routine de chargement d'un octet), donc aucune vérification n'est réellement faite

-Aucune vérification n'est faite que l'entête s'est bien chargé (parité). Si tel n'est pas le cas, le programme mal enregistré se charge n'importe où, créant des catastrophes...

```
E7AA  LDA 9A      ..............
E7AC  LDY 9B      .............. prendre début Basic
E7AE  STA 5F      .............. comme début par défaut
E7B0  STY 60      .............. (complètement inutile)
E7B2  PHP         .............. sauver P (indicateur d'interruption)
E7B3  JSR $E725   .............. analyser la syntaxe
E7B6  JSR $E6CA   .............. configurer le VIA pour E/S
E7B9  JSR $E4A8   .............. charger le programme
E7BC  JSR $E804   .............. reconfigurer le VIA
E7BF  PLP         .............. récupérer IRQ
E7C0  LDX 61      ..............
```

```
E7C2  LDA 62      ..............  prendre fin du programme
E7C4  STA 9D      ..............
E7C6  STX 9C      ..............  comme fin du programme Basic (!)
E7C8  LDA 63      ..............  mode AUTO ?
E7CA  BEQ E7D6    ..............  non, retour Basic (!)
E7CC  LDA 64      ..............  oui, Basic ?
E7CE  BEQ E7D3    ..............   oui, sauter
E7D0  JMP (5F)    ..............   non, exécution au début
E7D3  JMP $C98B   ..............   faire 'RUN'
E7D6  PLA         ..............
E7D7  PLA         ..............  enlever adresser de retour
E7D8  JMP $C96B   ..............  et direct à l'interpréteur
```

Remarque: cette fois, la routine n'est plus boguée, c'est nettement mieux !

```
..............  E85B  PHP         sauver P (I surtout)
..............  E85C  JSR $E7B2   analyser la syntaxe
..............  E85F  LDA 02AD    si AUTO
..............  E862  ORA 02AE    ou A ou E
..............  E865  BNE E871    alors SYNTAX ERROR
..............  E867  LDA 025A    J ?
..............  E86A  BEQ E874    non, sauter
..............  E86C  LDA 025B    J et V en même temps ?
..............  E86F  BEQ E874    non, c'est bon
..............  E871  JMP $D070   oui, 'SYNTAX ERROR'
..............  E874  JSR $E76A   configurer VIA
..............  E877  JSR $E57D   afficher 'Searching'
..............  E87A  JSR $E4AC   chercher le programme demandé
..............  E87D  BIT 02AE    Tableau ?
..............  E880  BVS E87A    oui, chercher un autre programme
..............  E882  LDA 025A    merge demandé ?
..............  E885  BEQ E8B3    non, sauter
```

Traiter J(oin)

```
..............  E887  LDA 02AE    est-ce un bloc mémoire ?
..............  E88A  BNE E87A    oui, chercher un autre programme
..............  E88C  LDA 9C
..............  E88E  LDY 9D      prendre fin du Basic
..............  E890  SEC
..............  E891  SBC #02
..............  E893  BCS E896
..............  E895  DEY         et ajuster
..............  E896  STA 02A9
```

```
..... E899 STY Ø2AA    comme nouveau début du programme
..... E89C SEC
..... E89D SBC 9A
..... E89F TAX         calculer dans XY la longueur du programme
..... E8AØ TYA         Basic en mémoire
..... E8A1 SBC 9B
..... E8A3 TAY
..... E8A4 CLC
..... E8A5 TXA         et ajouter à fin du programme trouvé
..... E8A6 ADC Ø2AB
..... E8A9 STA Ø2AB    ce qui donne la nouvelle fin
..... E8AC TYA
..... E8AD ADC Ø2AC
..... E8BØ STA Ø2AC

..... E8B3 JSR $E59B   afficher 'Loading/Verifying'
..... E8B6 JSR $E4EØ   charger/vérifier le programme
..... E8B9 JSR $E93D   reconfigurer le VIA
..... E8BC PLP         récupérer I
..... E8BD LDA Ø25B    c'était une vérification ?
..... E8CØ BEQ E8D3    non, sauter
..... E8C2 LDX Ø25C    oui, afficher le nombre d'erreurs trouvées
..... E8C5 LDA Ø25D
..... E8C8 JSR $EØC5
..... E8CB LDA #52
..... E8CD LDY #E5     indexer 'Verify errors detected'
..... E8CF JSR $CCBØ   et afficher
..... E8D2 RTS

..... E8D3 JSR $E651   vérifier pas d'erreur de parité
..... E8D6 LDA Ø2AE    c'était du BASIC ?
..... E8D9 BEQ E8E9    oui, sauter
..... E8DB LDA Ø2AD    AUTO ?
..... E8DE BEQ E8E8    non, on sort
..... E8EØ LDA Ø2B1    tester si erreur détectée
..... E8E3 NOP         (la version précédente comportait
..... E8E4 NOP         un BNE E8E8 qui empêchait l'exécution
..... E8E5 JMP (Ø2A9)  et exécuter le programme
..... E8E8 RTS
..... E8E9 LDX Ø2AB
..... E8EC LDA Ø2AC    fin du programme
..... E8EF STX 9C
..... E8F1 STA 9D      comme fin du texte Basic
..... E8F3 JSR $C55F   ajuster les liens
```

```
.....  E8F6  LDA Ø2AD    AUTO ?
.....  E8F9  BEQ E9Ø3    non,sortir simplement
.....  E8FB  LDA Ø2B1    prendre indicateur d'erreur de parité
.....  E8FE  NOP
.....  E8FF  NOP         anciennement BNE E9Ø3
.....  E9ØØ  JMP $C7Ø8   CLEAR, TXTPTR=début du programme
.....  E9Ø3  JSR $C7Ø8   faire CLEAR
.....  E9Ø6  JMP $C4A8   et directement à l'interpréteur
```

'CSAVE' (COMMANDE)

```
E7DB  LDA 9A      E9Ø9  LDA 9A
E7DD  LDY 9B      E9ØB  LDY 9B      prendre début du texte Basic
E7DF  STA 5F      E9ØD  STA Ø2A9
E7E1  STY 6Ø      E91Ø  STY Ø2AA    comme début du programme par défaut
E7E3  LDA 9C      E913  LDA 9C
E7E5  LDY 9D      E915  LDY 9D      et fin du texte
E7E7  STA 61      E917  STA Ø2AB
E7E9  STY 62      E91A  STY Ø2AC    comme fin du programme par défaut
E7EB  PHP         E91D  PHP         sauver I surtout
E7EC  JSR $E725   E91E  JSR $E7B2   analyser la syntaxe
.....             E921  LDA Ø25A    si J
.....             E924  ORA Ø25B    ou V demandé,
.....             E927  BEQ E92C
.....             E929  JMP $DØ7Ø   alors 'SYNTAX ERROR'
E7EF  JSR $E6CA   E92C  JSR $E76A   configurer le VIA
.....             E92F  JSR $E585   afficher 'Saving ..' et le nom du programme
.....             E932  JSR $E6Ø7   sauver l'entête
E7F2  JSR $E57B   E935  JSR $E62E   sauver le programme
E7F5  JSR $E8Ø4   E938  JSR $E93D   reconfigurer le VIA
E7F8  PLP         E93B  PLP         récupérer I
E7F9  LDX A9      .....             mode direct ?
E7FB  INX         .....
E7FC  BEQ E7FF    .....             oui, retour interpréteur
E7FE  RTS         E93C  RTS         retour
E7FF  PLA         .....
E8ØØ  PLA         .....             ajuster la pile
E8Ø1  JMP $C96B   .....             et saut á l'interpréteur
```

RECONFIGURER LE VIA

```
E8Ø4  JSR $E563   E93D  JSR $E5F5   effacer la ligne Ø
```

```
E8Ø7  JSR $F439   E94Ø  JSR $F9AA   initialiser le VIA
E8ØA  JMP $EEDØ   E943  JMP $EDEØ   et autoriser les IRQ par T1
```

'CALL' (COMMANDE)

```
E8ØD  JSR $E79D   E946  JSR $E853   évaluer l'adresse
E81Ø  JMP (ØØ33)  E949  JMP (ØØ33)  et exécution...
```

EVALUER UN NOMBRE HEXA

Entrée: TXTPTR pointe sur le '#'

Sortie: YA=valeur (aussi #33-#34 pour la V1.Ø et #ØC-#ØD pour la V1.1)

Bogue: la V1.Ø utilise #33-#34 comme pointeur de travail, qui est aussi utilisée par le POKE pour sauver l'adresse du POKE, d'où divers problèmes pour les POKE d'un chiffre HEXA...

```
E813  LDX #ØØ     E94C  LDX #ØØ
E815  STX 33      E94E  STX ØC
E817  STX 34      E95Ø  STX ØD      vider la zone résultat
E819  BEQ E82E    E952  BEQ E967    inconditionnel
```

Assembler un chiffre

```
E81B  LDX #Ø3     E954  LDX #Ø3     indiquer il faut 4 bits
E81D  ASL A       E956  ASL A       déplacer bØ-b1-b2-b3
E81E  ASL A       E957  ASL A
E81F  ASL A       E958  ASL A
E82Ø  ASL A       E959  ASL A       vers b4-b5-b6-b7
E821  ASL A       E95A  ASL A       sortir un bit
E822  ROL 33      E95B  ROL ØC      et répercuter dans le poids faible
E824  ROL 34      E95D  ROL ØD      et dans le poids fort...
E826  BCC E82B    E95F  BCC E964   s'il sort un 1,
E828  JMP $DBEØ   E961  JMP $DC39   alors 'OVERFLOW ERROR'
E82B  DEX         E964  DEX         4 bits assemblés ?
E82C  BPL E821    E965  BPL E95A   non, on continue
E82E  JSR $ØØE2   E967  JSR $ØØE2   prendre caractère suivant
E831  CMP #8Ø     E96A  CMP #8Ø     est-ce un mot clé Basic ?
E833  BCS E843    E96C  BCS E97C   oui, on sort
E835  ORA #8Ø     E96E  ORA #8Ø     b7=1 (inutile, le EOR aurait pu compenser)
E837  EOR #BØ     E97Ø  EOR #BØ     b7=Ø et enlever code pour 'Ø' (#3Ø)
```

```
E839  CMP #ØA     E972  CMP #ØA     il reste moins de 1Ø ?
E83B  BCC E81B    E974  BCC E954    oui, assembler le quartet
E83D  ADC #88     E976  ADC #88     ajouter pour compenser si lettre
E83F  CMP #FA     E978  CMP #FA     est-ce supérieur à 9 ?
E841  BCS E81B    E97A  BCS E954    oui, assembler
E843  LDA 34      E97C  LDA ØD      sortir
E845  LDY 33      E97E  LDY ØC      YA=valeur de l'octet
E847  RTS         E98Ø  RTS
```

PRENDRE UN NOMBRE HEXA DANS ACC1

```
E848  JSR $E813   E981  JSR $E94C   prendre un nombre hexa dans YA
E84B  JMP $D8D5   E984  JMP $DF4Ø   YA --) ACC1
```

'STORE' (COMMANDE)

```
.....  E987  PHP          sauver I
.....  E988  JSR $EA57    analyse de syntaxe
.....  E98B  LDA #4Ø      indiquer
.....  E98D  STA Ø2AE     programme de type tableau...
.....  E99Ø  LDA 28
.....  E992  STA Ø2AF     sauver drapeau réel/chaine
.....  E995  LDA 29
.....  E997  STA Ø2BØ     et drapeau entier/réel
.....  E99A  JSR $E585    afficher 'Saving ..' et le nom du programme
.....  E99D  JSR $E6Ø7    sauver l'entête du programme
.....  E9AØ  JSR $EA9E    préparer pour sauver tableau
.....  E9A3  JSR $E62E    et sauver sur la cassette
.....  E9A6  BIT 28       chaine ou nombre ?
.....  E9A8  BPL E9CC     nombre:on sort...
.....  E9AA  LDY #ØØ      indexer la longueur d la chaine
.....  E9AC  LDA (ØC),Y   prendre la longueur
.....  E9AE  BEQ E9C7     si chaine vide, élément suivant
.....  E9BØ  TAX          longueur dans X
.....  E9B1  LDY #Ø2
.....  E9B3  LDA (ØC),Y
.....  E9B5  STA ØØDØ,Y   sinon, mettre dans #D1-#D2
.....  E9B8  DEY          l'adresse de la chaine
.....  E9B9  BNE E9B3
.....  E9BB  INX          ajuster la longueur
.....  E9BC  DEX          et décompter caractères sauvés
.....  E9BD  BEQ E9C7     c'est fini, chaine suivante
```

```
..... E9BF  LDA (D1),Y  on prend le caractère
..... E9C1  JSR $E65E   et on le sauve
..... E9C4  INY         on indexe le caractère suivant
..... E9C5  BNE E9BC    inconditionnel: continuer
..... E9C7  JSR $EA42   suivant et test de fin
..... E9CA  BCC E9AA    non, chaine suivante
..... E9CC  JSR $E93D   oui, reconfigurer le VIA
..... E9CF  PLP         récupérer I
..... E9D0  RTS
```

'RECALL' (COMMANDE)

Bogue: pour un tableau de chaines, un 'OUT OF MEMORY' risque d'être déclenché s'il n'y a pas assez de place pour les chaines, laissant ainsi le clavier bloqué. Il est vrai qu'il n'y a quasiment pas de solution pour l'éviter

```
..... E9D1  JSR $D650   réorganiser les chaines
..... E9D4  PHP         sauver I
..... E9D5  JSR $EA57   analyser la syntaxe
..... E9D8  JSR $E57D   afficher 'Searching ..'
..... E9DB  JSR $E4AC   et chercher le programme
..... E9DE  BIT 02AE    et tester que c'est bien un tableau
..... E9E1  BVC E9DB    non, on recherche un autre programme
..... E9E3  LDA 02AF    oui, on vérifier même type
..... E9E6  EOR 28      pour le drapeau chaine/nombre
..... E9E8  BNE E9DB    non, on cherche autre chose
..... E9EA  LDA 02B0    et pour le drapeau entier/réel
..... E9ED  EOR 29
..... E9EF  BNE E9DB    non, on cherche autre chose
..... E9F1  JSR $E59B   afficher 'Loading' et le nom du programme
..... E9F4  LDY #02
..... E9F6  LDA (CE),Y  faire pseudo différence de
..... E9F8  CMP 02A9    la longueur des tableaux
..... E9FB  INY         pour vérifier dimensionnement
..... E9FC  LDA (CE),Y
..... E9FE  SBC 02AA
..... EA01  BCS EA09
..... EA03  JSR $E93D   dimensionement incorrect,on replace le VIA
..... EA06  JMP $C47C   et 'OUT OF MEMORY ERROR'
..... EA09  JSR $EA9E   calculer le début du tableau
..... EA0C  JSR $E4E0   et charger le tableau à l'adresse calculée
..... EA0F  BIT 28      est-ce un tableau numérique ?
..... EA11  BPL EA3A    oui, on sort
```

```
..............  EA13  LDY #ØØ      non, indexer longueur de la chaine
..............  EA15  LDA (ØC),Y   et prendre la longueur
..............  EA17  BEQ EA35     la chaine est vide, chaine suivante
..............  EA19  JSR $D5AB    sinon, réserver la place pour la chaine
..............  EA1C  LDY #ØØ      indexer le début de la chaine
..............  EA1E  TAX          longueur dans X
..............  EA1F  INX          ajuster la longueur
..............  EA2Ø  DEX          et décompter les caractères de la chaine
..............  EA21  BEQ EA2B     chaine suivante si tous transférés
..............  EA23  JSR $E6C9    lire un caractère sur la cassette
..............  EA26  STA (D1),Y   et sauver en mémoire
..............  EA28  INY          préparer pour caractère suivant
..............  EA29  BNE EA2Ø     inconditionnel
..............  EA2B  LDY #Ø2
..............  EA2D  LDA ØØDØ,Y   prendre adresse de la chaine
..............  EA3Ø  STA (ØC),Y   et transfert vers le tableau
..............  EA32  DEY
..............  EA33  BNE EA2D
..............  EA35  JSR $EA42    prendre descripteur suivant
..............  EA38  BCC EA13     si pas encore la fin, on continue
..............  EA3A  JSR $E93D    reconfigurer le VIA
..............  EA3D  JSR $E651    afficher éventuellement 'Errors found'
..............  EA4Ø  PLP          récupérer I
..............  EA41  RTS
```

DESCRIPTEUR SUIVANT

```
..............  EA42  CLC
..............  EA43  LDA #Ø3
..............  EA45  ADC ØC       ajouter 3 à l'adresse courante
..............  EA47  STA ØC
..............  EA49  BCC EA4D
..............  EA4B  INC ØD       répercution dans le poids fort...
..............  EA4D  TAY          poids faible dans Y
..............  EA4E  LDA ØD
..............  EA5Ø  CPY Ø2AB     faire pseudo différence
..............  EA53  SBC Ø2AC     pour calculer si la fin est atteinte
..............  EA56  RTS
```

SYNTAXE STORE/RECALL

```
..............  EA57  LDA #4Ø
```

```
..............  EA59  STA 2B      indiquer que l'on veut le nom
..............  EA5B  JSR $D188   et prendre adresse du tableau
..............  EA5E  LDA #ØØ     replacer correctement l'indicateur
..............  EA6Ø  STA 2B
..............  EA62  LDY #Ø3
..............  EA64  LDA (CE),Y  prendre longueur du tableau poids fort
..............  EA66  STA Ø2AA    et sauver temporairement
..............  EA69  DEY
..............  EA6A  LDA (CE),Y
..............  EA6C  STA Ø2A9    idem poids faible
..............  EA6F  BNE EA74    et décrémenter
..............  EA71  DEC Ø2AA
..............  EA74  DEC Ø2A9
..............  EA77  JSR $DØ65   demander ','
..............  EA7A  LDA 29      sauver type du tableau
..............  EA7C  PHA
..............  EA7D  LDA 28
..............  EA7F  PHA
..............  EA8Ø  JSR $E7B2   prendre le nom du programme
..............  EA83  PLA
..............  EA84  STA 28
..............  EA86  PLA
..............  EA87  STA 29      récupérer le type du tableau
..............  EA89  LDA Ø25B    si ,V
..............  EA8C  ORA Ø2AD    ou AUTO
..............  EA8F  ORA Ø2AE    ou ,A ou ,E
..............  EA92  ORA Ø25A    ou encore ,J
..............  EA95  BEQ EA9A
..............  EA97  JMP $DØ7Ø   alors 'SYNTAX ERROR'
..............  EA9A  JSR $E76A   sinon, configurer le VIA
..............  EA9D  RTS
```

AJUSTER ADRESSE DU TABLEAU

```
..............  EA9E  CLC
..............  EA9F  LDA CE      prendre adresse du début
..............  EAA1  ADC Ø2A9    ajouter la longueur poids faible
..............  EAA4  STA Ø2AB    et sauver la fin du tableau
..............  EAA7  LDA CF
..............  EAA9  ADC Ø2AA
..............  EAAC  STA Ø2AC    idem poids fort
..............  EAAF  LDY #Ø4     indexer le nombre de dimensions
..............  EAB1  LDA (CE),Y  et prendre la dimension du tableau
```

```
.............. EAB3  JSR $D288   calculer début effectif du tableau
.............. EAB6  STA Ø2A9
.............. EAB9  STY Ø2AA    comme vraie adresse de début
.............. EABC  STA ØC
.............. EABE  STY ØD      et comme pointeur des  escripteurs
.............. EACØ  RTS
```

6) CONGIGURATION PAR LE BASIC

1-Listing

TABLE DE DONNEES 'SOUND' à 'INK'

Table d'adresse

(Les adresses sont en fait stockées avec leur valeur-1, poids faible d'abord)

```
E84E ....  BYT #F41D,#F423,#F42Ø
E854 ....  BYT #EBDE,#EBE1,#EBE4
E85A ....  BYT #EBE7,#EBEA,#EBF9
E86Ø ....  BYT #EBED,#EBF3,#EBF6
.... EAC1  BYT #FB3F,#FC17,#FBCF
.... EAC7  BYT #FØC7,#FØFC,#F1ØF
.... EACD  BYT #F37E,#F11C,#F267
.... EAD3  BYT #F12C,#F2Ø3,#F2ØF
```

Table du nombre de paramètres

```
E866 EAD9  BYT 3,4,4,3,3,3,2,1,3,3,1,1
```

Table d'adressage du curseur

```
E872 EAE5  BYT Ø,Ø,Ø,Ø,1,1,Ø,Ø,Ø,Ø,Ø,Ø
```

INTERPRETATION GRAPHIQUE

```
E87D  LDA Ø2CØ    EAFØ  LDA Ø2CØ    prendre indicateur de mode
E88Ø  AND #Ø1     EAF3  AND #Ø1     et isoler le bit HIRES/TEXT
E882  BNE E889    EAF5  BNE EAFC    (LSR/BCS aurait été mieux)
```

```
E884  LDX #A3     EAF7  LDX #A3     si pas mode HIRES,'DISP TYPE MISMATCH'
E886  JMP $C485   EAF9  JMP $C47E
```

INTERPRETATION SON

```
E889  CPY #4E     EAFC  CPY #4E     le mot-clé est-il bien entre 'SOUND'
E88B  BCS E890    EAFE  BCS EB03
E88D  JMP $CFE4   EB00  JMP $D07F   (vérification inutile)
E890  CPY #66     EB03  CPY #66     et 'INK' ?
E892  BCS E88D    EB05  BCS EB00    non, 'SYNTAX ERROR'
E894  TYA         EB07  TYA         oui, index dans A
E895  SEC         EB08  SEC         et ramener à 0-22
E896  SBC #4E     EB09  SBC #4E     (inutile, il suffit de décaler l'adresse
E898  TAY         EB0B  TAY         de base de #4E...)
E899  LDA E84F,Y  EB0C  LDA EAC2,Y  prendre adresse d'exécution poids fort
E89C  PHA         EB0F  PHA         et on sauve sur la pile
E89D  LDA E84E,Y  EB10  LDA EAC1,Y  poids faible aussi
E8A0  PHA         EB13  PHA         pour exécution par RTS
E8A1  TYA         EB14  TYA         ramener index à 0-11
E8A2  LSR A       EB15  LSR A       il aurait été plus judicieux d'entrelacer
E8A3  TAY         EB16  TAY         les deux tables
E8A4  LDA E866,Y  EB17  LDA EAD9,Y  prendre nombre de paramètres
E8A7  PHA         EB1A  PHA         et on sauve
E8A8  LDA E872,Y  EB1B  LDA EAE5,Y  prendre adressage curseur
E8AB  STA 02C3    EB1E  STA 02C3    on sauve
E8AE  LDA #00     EB21  LDA #00     initialiser compteur
E8B0  STA 02F0    EB23  STA 02F0    de paramètres pris
E8B3  JSR $CE77   EB26  JSR $CF03   évaluer une expression numérique
E8B6  LDA 02C3    EB29  LDA 02C3    mode relatif ?
E8B9  BNE E8C1    EB2C  BNE EB34    oui, on saute
E8BB  JSR $D867   EB2E  JSR $D922   ACC1 --> #33-#34 non signé
E8BE  JMP $E8C8   EB31  JMP $EB3B   et on continue
E8C1  LDA D0      EB34  LDA D0      prendre exposant
E8C3  CMP #90     EB36  CMP #90     et vérifier inférieur à 15
E8C5  JSR $D86F   EB38  JSR $D92A   et convertir ACC1 dans #33-#34
E8C8  LDY 02F0    EB3B  LDY 02F0    prendre numéro du paramètre en cours
E8CB  LDA 33      EB3E  LDA 33
E8CD  STA 02E1,Y  EB40  STA 02E1,Y  sauver poids faible
E8D0  LDA 34      EB43  LDA 34
E8D2  STA 02E2,Y  EB45  STA 02E2,Y  et idem poids fort
E8D5  INY         EB48  INY
E8D6  INY         EB49  INY         préparer paramètre suivant
E8D7  STY 02F0    EB4A  STY 02F0    et sauver paramètre en cours
```

```
E8DA  PLA          EB4D  PLA          récupérer nombre de paramètres
E8DB  TAY          EB4E  TAY          dans Y
E8DC  DEY          EB4F  DEY          on compte
E8DD  BE@ E8E7     EB5Ø  BE@ EB5A     si c'est fini, on sort
E8DF  TYA          EB52  TYA          sinon on réempile
E8EØ  PHA          EB53  PHA
E8E1  JSR $CFD9    EB54  JSR $DØ65    on demande une virgule
E8E4  JMP $E8B3    EB57  JMP $EB26    et on recommence
E8E7  LDA #ØØ      EB5A  LDA #ØØ      indiquer pas d'erreur pour l'instant
E8E9  STA Ø2EØ     EB5C  STA Ø2EØ
E8EC  PLA          EB5F  PLA
E8ED  TAX          EB6Ø  TAX          récupérer adresse de la commande
E8EE  PLA          EB61  PLA
E8EF  TAY          EB62  TAY          dans XY
E8FØ  LDA #E8      EB63  LDA #EB      pour empiler l'adresse de retour
E8F2  PHA          EB65  PHA
E8F3  LDA #FA      EB66  LDA #6D      soit #E8FA/#EB6D
E8F5  PHA          EB68  PHA
E8F6  TYA          EB69  TYA          et empiler à nouveau
E8F7  PHA          EB6A  PHA          adresse de la commande.
E8F8  TXA          EB6B  TXA          il aurait été si simple d'empiler l'adresse
E8F9  PHA          EB6C  PHA          de retour avant !
E8FA  RTS          EB6D  RTS
```

RETOUR COMMANDES GRAPHIQUES ET SONORES

Remarque: toutes ls commandes graphiques et sonores passent ici au retour, puisque cette adresse est empilée, simulant un appel par JSR.

```
E8FB  LDA #Ø1      EB6E  LDA #Ø1      tester indicateur d'erreur
E8FD  BIT Ø2EØ     EB7Ø  BIT Ø2EØ     (LDA Ø2EØ/BE@ aurait marché
E9ØØ  BE@ E8FA     EB73  BE@ EB6D     pas d'erreur, on sort
E9Ø2  JMP $D2AØ    EB75  JMP $D336    'ILLEGAL QUANTITY ERROR'
```

GERER LE TAMPON TOUCHE

Entrée: les IR@ doivent @tre autorisées, sinon il ne se passera rien !

Sortie: N=Ø si pas de touche frappée
N=1 et A=code ASCII, #Ø2DF=Ø si une touche a été frappée.
Dans tous les cas, X et Y ne sont pas affectés.

Remarque: normalement, si N=Ø (pas de touche), le tampon et donc A vaut aussi Ø. C'est toujours vrai, sauf après une interruption de LIST par une touche, car le tampon est vidé par LSR Ø2DF, et non LDA #ØØ/STA Ø2DF, ce qui est d'ailleurs aussi bien.

```
E9Ø5  LDA Ø2DF    EB78  LDA Ø2DF    prendre le tampon touche
E9Ø8  BPL E915    EB7B  BPL EB88    on sort N=Ø s'il est vide
E9ØA  PHP         EB7D  PHP         sauver b7=1 (soit N)
E9ØB  AND #7F     EB7E  AND #7F     éliminer b7=1
E9ØD  PHA         EB8Ø  PHA         et sauver
E9ØE  LDA #ØØ     EB81  LDA #ØØ     vider le tampon
E91Ø  STA Ø2DF    EB83  STA Ø2DF    (LSR Ø2DF aurait évité de sauver A)
E913  PLA         EB86  PLA         récupérer code ASCII
E914  PLP         EB87  PLP         et placer N=1
E915  RTS         EB88  RTS
```

VERIFIER AY

Entrée: AY contient la valeur du HIMEM désiré

Sortie: C=1 si AY n'est pas correct compte tenu du mode et du programme en mémoire.

C=Ø si tut est Ok.

```
E916  CPY 9D      EB89  CPY 9D      tester si au dessus programme
E918  BCS E91C    EB8B  BCS EB8F
E91A  SEC         EB8D  SEC         non, on sort, C=1
E91B  RTS         EB8E  RTS
E91C  BNE E924    EB8F  BNE EB97
E91E  CMP 9C      EB91  CMP 9C      pas sur, tester poids faible
E92Ø  BCC E91B    EB93  BCC EB8E
E922  BEQ E91B    EB95  BEQ EB8E
E924  JSR $E942   EB97  JSR $EBB5   tester par rapport au maxi mode RELEASE
E927  BCC E91B    EB9A  BCC EB8E    en dessous, c'est bon, C=Ø
E929  TAX         EB9C  TAX         sauver poids faible
E92A  LDA Ø2CØ    EB9D  LDA Ø2CØ    prendre indicateur de mode
E92D  AND #Ø2     EBAØ  AND #Ø2     et tester si mode GRAB
E92F  PHP         EBA2  PHP         sauver le mode dans Z
E93Ø  TXA         EBA3  TXA         reprendre le poids faible
E931  PLP         EBA4  PLP         et le mode
E932  BNE E91A    EBA5  BNE EB8D    si mode RELEASE, on sort C=1
E934  TYA         EBA7  TYA         sinon, on va voir si pas trop haut
```

```
E935  PHA          EBA8  PHA          sauver poids fort
E936  SEC          EBA9  SEC          retrancher #1C00
E937  SBC #1C      EBAA  SBC #1C
E939  TAY          EBAC  TAY          dans Y
E93A  TXA          EBAD  TXA          récupérer poids faible
E93B  JSR $E942    EBAE  JSR $EBB5   et tester si au dessous maxi RELEASE
E93E  PLA          EBB1  PLA          (le résultat est dans C)
E93F  TAY          EBB2  TAY          récupérer vrai poids fort
E940  TXA          EBB3  TXA          inutile
E941  RTS          EBB4  RTS

Verifier AY au dessous maxi RELEASE (#9800 ou #1800 pour un 16 K)

E942  CPY 02C2     EBB5  CPY 02C2     tester poids fort
E945  BCC E949     EBB8  BCC EBBC     si en dessous,sortir C=0
E947  BEQ E94A     EBBA  BEQ EBBD     si égal,tester poids faible ausssi
E949  RTS          EBBC  RTS          sinon, C=1
E94A  CMP 02C1     EBBD  CMP 02C1     tester poids faible
E94D  RTS          EBC0  RTS

Prendre dans AY maxi RELEASE -1

E94E  LDY 02C2     EBC1  LDY 02C2     prendre poids fort
E951  LDA 02C1     EBC4  LDA 02C1     et poids faible
E954  BNE E957     EBC7  BNE EBCA     si non nul,pas toucher au poids fort
E956  DEY          EBC9  DEY          décrémenter poids fort
E957  SEC          EBCA  SEC          et enlever en tous cas 1 au poids faible
E958  SBC #01      EBCB  SBC #01
E95A  RTS          EBCD  RTS
```

'HIMEM' (COMMANDE)

```
Utilisation: entrée en #E965/#EBD8 avec la valeur désirée dans AY.

E95B  JSR $CE77    EBCE  JSR $CF03
E95E  JSR $D867    EBD1  JSR $D922     prendre un entier dans YA et #33-#34
E961  LDA 33       EBD4  LDA 33
E963  LDY 34       EBD6  LDY 34
E965  JSR $E916    EBD8  JSR $EB89     vérifier validité de l'adresse donnée
E968  BCC E96D     EBDB  BCC EBE0
E96A  JMP $C483    EBDD  JMP $C47C     'OUT OF MEMORY ERROR'
E96D  STA A6       EBE0  STA A6
E96F  STY A7       EBE2  STY A7        sauver valeur HIMEM
E971  JMP $C73A    EBE4  JMP $C70F     et faire CLEAR
```

'GRAB' (COMMANDE)

```
.............. EBE7 LDA Ø26Ø
.............. EBEA BNE EBDD
E974 LDA Ø2CØ  EBEC LDA Ø2CØ   prendre indicateur de mode
E977 PHA       EBEF PHA        sauver
E978 AND #Ø1   EBFØ AND #Ø1    tester pour HIRES
E97A BEQ E981  EBF2 BEQ EBF9
E97C LDX #A3   EBF4 LDX #A3    oui, 'DISP TYPE MISMATCH' (3 ème fois !)
E97E JMP $C485 EBF6 JMP $C47E
E981 PLA       EBF9 PLA        récupérer mode
E982 AND #FD   EBFA AND #FD    indiquer mode GRAB (b1=Ø)
E984 STA Ø2CØ  EBFC STA Ø2CØ   et resauver
E987 JSR $E94E EBFF JSR $EBC1  prendre top mémoire (#2C1-#2C2)
E98A PHA       EC02 PHA        sauver poids faible
E98B TYA       EC03 TYA
E98C CLC       EC04 CLC        poids fort dans A
E98D ADC #1C   EC05 ADC #1C    ajouter #1CØØ (#B4ØØ-#98ØØ)
E98F TAY       EC07 TAY        dans Y poids fort
E99Ø PLA       EC08 PLA        et A poids faible
E991 JMP $E96D EC09 JMP $EBEØ  et finir comme HIMEM
```

'RELEASE' (COMMANDE)

```
E994 JSR $E94E EC0C JSR $EBC1  prendre top mémoire
E997 JSR $E916 EC0F JSR $EB89  vérifier possible
E99A BCS E96A  EC12 BCS EBDD   non, erreur
E99C PHA       EC14 PHA        sauver poids faible
E99D LDA Ø2CØ  EC15 LDA Ø2CØ   prendre indicateur de mode
E9AØ ORA #Ø2   EC18 ORA #Ø2    indiquer RELEASE (b1=1)
E9A2 STA Ø2CØ  EC1A STA Ø2CØ   et resauver
E9A5 PLA       EC1D PLA        récupérer poids faible
E9A6 JMP $E96D EC1E JMP $EBEØ  et finir comme HIMEM
```

'TEXT' (COMMANDE)

Remarque: cette commande peut être utilisée telle quelle en assembleur.

```
E9A9 LDA Ø2CØ  EC21 LDA Ø2CØ   prendre indicateur de mode
E9AC TAY       EC24 TAY        le sauver
E9AD AND #Ø1   EC25 AND #Ø1    et tester si déjà mode TEXT
E9AF BEQ E9BA  EC27 BEQ EC32   oui, on sort
```

```
E9B1  TYA          EC29  TYA          récupérer le mode
E9B2  AND #FE      EC2A  AND #FE      indiquer mode TEXT (b0=0)
E9B4  STA 02C0     EC2C  STA 02C0     et resauver
E9B7  JSR $F427    EC2F  JSR $F967    configurer l'écran
E9BA  RTS          EC32  RTS
```

'HIRES' (COMMANDE)

Remarque: cette commande peut être utilisée telle quelle en assembleur.

```
E9BB  LDA 02C0     EC33  LDA 02C0     prendre indicateur de mode
E9BE  PHA          EC36  PHA          sauver
E9BF  AND #02      EC37  AND #02      tester si GRAB
E9C1  BEQ E97C     EC39  BEQ EBF4     oui, 'DISP TYPE MISMATCH ERROR'
E9C3  PLA          EC3B  PLA          récupérer mode
E9C4  ORA #01      EC3C  ORA #01      indiqur HIRES (b0=1)
E9C6  STA 02C0     EC3E  STA 02C0     et resauver
E9C9  JSR $F42A    EC41  JSR $F920    configurer l'écran
E9CC  RTS          EC44  RTS
```

'POINT' (FONCTION)

Bogue: au lieu d'utiliser la #CE77/CF03, qui vérifie que le paramètre est bien numérique, la #CE8B/#CF17 est utilisée, qui accepte aussi bien les chaines.

Donc,les chaines sont acceptées comme argument, ce qui ne veut rien dire (POINT ("X","Y") est accepté).

Dans ce cas,le réservation de chaine n'est pas enlevée, et il apparait très vite un FORMULA TOO COMPLEX ERROR car la pile des descripteurs n'est pas vidée. (C'est d'ailleurs un des rares moyens simple d'obtenir ce message)

```
E9CD  JSR $CFD6    EC45  JSR $D062    demander '('
E9D0  JSR $CE8B    EC48  JSR $CF17    évaluer une expression
E9D3  LDA 34       EC4B  LDA 34
E9D5  PHA          EC4D  PHA          sauver #33-#34 (éviter problème avec POKE)
E9D6  LDA 33       EC4E  LDA 33
E9D8  PHA          EC50  PHA
E9D9  JSR $D867    EC51  JSR $D922    ACC1 --> #34-#33
E9DC  LDA 33       EC54  LDA 33       inutile, valeur dans YA...
E9DE  STA 02E1     EC56  STA 02E1     et sauver premiére coordonnée
E9E1  LDA 34       EC59  LDA 34
E9E3  STA 02E2     EC5B  STA 02E2
E9E6  PLA          EC5E  PLA
```

```
E9E7  STA 33      EC5F  STA 33
E9E9  PLA         EC61  PLA
E9EA  STA 34      EC62  STA 34      inutile de récupérer pour resauver...
E9EC  JSR $CFD9   EC64  JSR $D0Ø65  demander ','
E9EF  JSR $CE8B   EC67  JSR $CF17   évaluer expression
E9F2  LDA 34      EC6A  LDA 34
E9F4  PHA         EC6C  PHA
E9F5  LDA 33      EC6D  LDA 33
E9F7  PHA         EC6F  PHA         sauver à nouveau #33-#34
E9F8  JSR $D867   EC7Ø  JSR $D922   ACC1 --) #33-#34
E9FB  LDA 34      EC73  LDA 34
E9FD  STA Ø2E4    EC75  STA Ø2E4
EAØØ  LDA 33      EC78  LDA 33
EAØ2  STA Ø2E3    EC7A  STA Ø2E3    et sauver deuxième coordonnée
EAØ5  PLA         EC7D  PLA
EAØ6  STA 33      EC7E  STA 33
EAØ8  PLA         EC8Ø  PLA
EAØ9  STA 34      EC81  STA 34      récupérer pour de bon #33-#34
EAØB  JSR $EBF1   EC83  JSR $F1C8   évaluer 'couleur' du point
EAØE  LDY Ø2E1    EC86  LDY Ø2E1    prendre résultat poids fort dans Y
EA11  LDA Ø2EØ    EC89  LDA Ø2EØ    au fait, erreur d'intervalle ?
EA14  AND #Ø1     EC8C  AND #Ø1     ...ou LSR
EA16  BNE EA21    EC8E  BNE EC99    oui, 'ILLEGAL QUANTITY ERROR
EA18  LDA Ø2E2    EC9Ø  LDA Ø2E2    prendre résultat poids faible
EA1B  JSR $D3ED   EC93  JSR $D499   YA --) ACC1 (signé) soit TRUE/FALSE
EA1E  JMP $CFD3   EC96  JMP $DØ5F   demander ')'
EA21  JMP $D8Ø7   EC99  JMP $D8C2   'ILLEGAL QUANTITY ERROR'
```

H) INITIALISATIONS BASIC

1-Listing

ROUTINE A TRANSFERER EN #E2

Programmation: compte tenu du manque de place en page Ø, la routine a été scindée en 2 morceaux. Toutefois,une fois de plus, il aurait été préférable de remplacer le JSR #EA41/#ECB9 par un JMP, économisant le RTS et surtout 9 µs à chaque appel de la routine.

On peut aussi évaluer exactement le temps perdu par un espace: 17 microsecondes.

```
EA24  INC E9      EC9C  INC E9      incrémenter TXTPTR
EA26  BNE EA2A    EC9E  BNE ECA2
EA28  INC EA      ECAØ  INC EA      éventuellement poids fort
EA2A  LDA EA6Ø    ECA2  LDA EA6Ø    prendre caractère dans le texte basic
EA2D  CMP #' '    ECA5  CMP #' '    est-ce un espace ?
EA2F  BEQ EA24    ECA7  BEQ EC9C   oui, ignorer
EA31  JSR $EA41   ECA9  JSR $ECB9   non, positionner Z et C
EA34  RTS         ECAC  RTS         et sortir
EA35  BIT EA6Ø    ECAD  BIT EA6Ø    ?
EA38  BIT EA6Ø    ECBØ  BIT EA6Ø    ??
EA3B  RTS         ECB3  RTS         ???
```

PREMIERE VALEUR DE RND

```
EA3C ECB4  BYT #8Ø,#4F,#C7,#52,#58  soit Ø,811635196
```

SUITE DE #E2

Entrée: A contient un code trouvé dans le programme

Sortie:Z=1 si NULL (#ØØ) ou ':'
C=Ø si chiffre (Ø à 9)

Attention:N n'est pas représentatif du caractère.

Remarque: excellente utilisation du code ASCII (':' est à la fois séparateur d'instruction et délimiteur de la zone des chiffres) Est-ce MICROSOFT qui s'est inspiré du code ASCII,ou le code qui s'est élaboré sur les conseils de MICROSOFT ?

Bogue: Les problèmes du ELSE et de la REM abrégées de la V1.1 viennent de cette routine: lorsqu'ils sont rencontrés, le saut à la routine qui ignore la fin de la ligne modifie Y et X, ce qui est déjà malsain, mais en plus, au retour A,Z et C ne sont pas représentatifs de la fin de la ligne, à cause d'un BNE #EA2A assez curieux !.

```
EA41  CMP &ELSE   ECB9  CMP &ELSE   est-ce ELSE ?
EA43  BNE EA4A    ..............   non, sauter
EA45  JSR $CA61   ..............   oui, ignorer via REM
EA48  BNE EA2A    ..............   inconditionnel: n'importe quoi !
..............    ECBB  BEQ ECCB   oui, on sort
EA4A  CMP #'''    ECBD  CMP #'''   est-ce REM abrégé ?
```

```
EA4C  BEQ EA45    ECBF  BEQ ECCB    oui, on sort ou on saute
EA4E  CMP #':'    ECC1  CMP #':'    est-ce le séparateur d'instructions ?
EA50  BCS EA58    ECC3  BCS ECCB    si au dessus, C=1 et de plus si égal, Z=1
EA52  SEC         ECC5  SEC
EA53  SBC #'0'    ECC6  SBC #'0'    convertir en -#30-#09
EA55  SEC         ECC8  SEC         et retrouver caractère original et placer Z
EA56  SBC #D0     ECC9  SBC #D0     avec dépassement (C=0) si chiffre
EA58  RTS         ECCB  RTS
```

RESET BASIC (COLD START BASIC)

```
EA59  CLD         ECCC  CLD         interdire mode décimal
EA5A  LDX #FF     ECCD  LDX #FF
EA5C  STX A9      ECCF  STX A9      mode direct
EA5E  TXS         ECD1  TXS         et initialiser la pile
EA5F  LDA #59     ECD2  LDA #CC
EA61  LDY #EA     ECD4  LDY #EC
EA63  STA 1B      ECD6  STA 1B      détourner le Ready en cas
EA65  STY 1C      ECD8  STY 1C      de NMI intempestive
EA67  LDA #4C     ECDA  LDA #4C     prendre code 6502 pour JMP
EA69  STA 1A      ECDC  STA 1A      vecteur Ready
EA6B  STA C3      ECDE  STA C3      vecteur exécution de fonctions
EA6D  STA 21      ECE0  STA 21      vecteur USR
EA6F  STA 02FB    ECE2  STA 02FB    vecteur &
EA72  LDA #A0     ECE5  LDA #36
EA74  LDY #D2     ECE7  LDY #D3     préparer 'ILLEGAL QUANTITY ERROR'
EA76  STA 22      ECE9  STA 22
EA78  STY 23      ECEB  STY 23      par défaut de USR
EA7A  STA 02FC    ECED  STA 02FC
EA7D  STY 02FD    ECF0  STY 02FD    par défaut de &
EA80  STA 02F5    ECF3  STA 02F5
EA83  STY 02F6    ECF6  STY 02F6    et par défaut de ! aussi
EA86  LDA #50     ..............
EA88  STA 31      ..............    longueur de ligne=80
EA8A  LDA #38     ..............
EA8C  STA 32      ..............    tabulation maximale à 56 (??)
EA8E  LDX #1C     ECF9  LDX #1C     transférer en #E2-#FE
EA90  LDA EA23,X  ECFB  LDA EC9B,X
EA93  STA E1,X    ECFE  STA E1,X    routine et valeur de RND
EA95  DEX         ED00  DEX
EA96  BNE EA90    ED01  BNE ECFB
EA98  LDA #03     ED03  LDA #03     établir taille normale d'un descripteur
EA9A  STA C2      ED05  STA C2      (voir la réorganisation des chaines)
```

```
EA9C  TXA          ED07  TXA          A=0 (??)
EA9D  STA D7       ED08  STA D7       justification ACC1=0 par défaut
EA9F  STA 87       ED0A  STA 87       poids fort pile des descripteurs=#00
EAA1  STA 2F       ED0C  STA 2F       ????
EAA3  PHA          ED0E  PHA          indiquer pas de blocs BASIC sur la pile
EAA4  STA 2E       ED0F  STA 2E       pas de Ctrl O
EAA6  STA 02F1     ..............     imprimante OFF
EA99  STA 02F2     ED11  STA 02F2     mode TROFF
EAAC  JSR $CC83    ..............     éviter vidéo inverse (!)
EAAF  JSR $CB9F    ..............     sauter une ligne
EAB2  LDX #88      ED14  LDX #88
EAB4  STX 85       ED16  STX 85       initialiser pile des descripteurs
EAB6  TAY          ED18  TAY          inutile
EAB7  LDA 02E1     ..............
EABA  LDY 02E2     ..............     prendre TOP de la mémoire
EABD  STA A6       ..............
EABF  STY A7       ..............     comme HIMEM provisoire
EAC1  LDA #02      ED19  LDA #02
EAC3  STA 02C0     ED1B  STA 02C0     Mode TEXT,RELEASE
EAC6  LDA #50      ..............
EAC8  STA 31       ..............     longueur ligne=80 (bis repetita !)
EACA  SBC #0E      ..............     et calculer position maxi
EACC  BCS EACA     ..............     de la tabulation
EACE  EOR #FF      ..............     pour une tabulation sur...14 colonnes !
EAD0  SBC #0C      ..............
EAD2  CLC          ..............
EAD3  ADC 31       ..............     utilité ?
EAD5  STA 32       ..............
EAD7  LSR 02F1     ..............     imprimante OFF (et de deux !)
..............     ED1E  LDA #28
..............     ED20  STA 0257     ligne écran=40
..............     ED23  LDA #50
..............     ED25  STA 0256     ligne imprimante=80
..............     ED28  LDA #00
..............     ED2A  STA 30       position tablulation=0
..............     ED2C  STA 0258     position écran aussi
..............     ED2F  STA 0259     de m@me que la position imprimante
EADA  JSR $CC83    ..............     mode NORMAL (2ème fois !?)
..............     ED32  JSR $C83E    initialiser #31 et #32,et sortie sur écran
EADD  JSR $CC0A    ED35  JSR $CCCE    effacer l'écran
EAE0  LDA #51      ED38  LDA #96
EAE2  LDY #EB      ED3A  LDY #ED      indexer 'Oric Extended basic...
EAE4  JSR $CBED    ED3C  JSR $CCB0    et afficher
EAE7  JSR $CB9F    ED3F  JSR $CBF0    sauter une ligne
```

```
EAEA  LDX #00      ED42  LDX #00
EAEC  LDY #05      ED44  LDY #05     début texte basic=#500
EAEE  STX 9A       ED46  STX 9A
EAF0  STY 9B       ED48  STY 9B
EAF2  LDY #00      ED4A  LDY #00
EAF4  TYA          ED4C  TYA
EAF5  STA (9A),Y   ED4D  STA (9A),Y  et y mettre un 0.Pourquoi pas STY #500 ?
EAF7  INC 9A       ED4F  INC 9A      ajuster sur #501
EAF9  BNE EAFD     ED51  BNE ED55
EAFB  INC 9B       ED53  INC 9B      répercuter sur le poids fort
EAFD  JSR $C71B    ED55  JSR $C6F0   faire NEW
EB00  LDA 9A       ED58  LDA 9A
EB02  LDY 9B       ED5A  LDY 9B
EB04  JSR $C448    ED5C  JSR $C444   vérifier mémoire disponible
EB07  JSR $CB9F    ED5F  JSR $CBF0   sauter une ligne
EB0A  LDA A6       ED62  LDA A6
EB0C  SEC          ED64  SEC
EB0D  SBC 9A       ED65  SBC 9A
EB0F  TAX          ED67  TAX         calculer dans XA la place disponible
EB10  LDA A7       ED68  LDA A7
EB12  SBC 9B       ED6A  SBC 9B
EB14  JSR $E0C1    ED6C  JSR $E0C5   et l'afficher
EB17  LDA #43      ED6F  LDA #88
EB19  LDY #EB      ED71  LDY #ED     indexer le message 'BYTE FREE'
EB1B  JSR $CBED    ED73  JSR $CCB0   et l'afficher
EB1E  SEC          ..............
EB1F  LDA A6       ..............
EB21  SBC #FF      ..............
EB23  STA A6       ..............    calculer le haut de la mémoire
EB25  STA 02C1     ..............    en mode RELEASE
EB28  LDA A7       ..............
EB2A  SBC #20      ..............
EB2C  STA A7       ..............    comme HIMEM
EB2E  STA 02C2     ..............    et sauver aussi maxi mode RELEASE
EB31  LDA #ED      ED76  LDA #B0
EB33  LDY #CB      ED78  LDY #CC
EB35  STA 1B       ED7A  STA 1B
EB37  STY 1C       ED7C  STY 1C      placer correctement vecteur Ready
EB39  LDA #10      ED7E  LDA #10     pourquoi pas 0 ou LSR #2F8 ?
EB3B  STA 02F8     ED80  STA 02F8    initialiser drapeau retour pour LIST
EB3E  JMP $C4B5    ED83  JMP $C4A8   et donner la main
```

MESSAGES D'INITIALISATION

```
EB41 ED86  BYT #ØØ,#ØØ
EB43 ED88  BYT ' BYTE FREE'
EB4E ED93  BYT #ØA,#ØD,#ØØ
EB51 ED96  BYT 'ORIC EXTENDED BASIC '
EB64 ....  BYT 'V1.Ø'
.... EDA9  BYT 'V1.1'
EB69 EDAE  BYT #ØD,#ØA,#6Ø
EB6C EDB1  BYT ' 1983 TANGERINE'
EB7B EDCØ  BYT #ØD,#ØA,#ØØ
```

Tout ce qui suit est inutilisé

```
EB7E ....  BYT 'TOO LARGE'
EB89 ....  BYT #ØD,#ØA,#ØØ,#ØØ
EB8D ....  BYT 'Software by PETER HALFORD AND ANDY BROWN'
EBB5 ....  BYT #ØØ

EBB6 ....  BYT #ØØ,#ØØ,#ØØ,#ØØ,#ØØ
EBBB ....  BYT #ØØ,#ØØ,#ØØ,#ØØ,#ØØ
EBCØ ....  BYT #FF,#FF,#FF,#FF
EBC4 ....  BYT #FF,#FF,#FF,#FF
EBC8 ....  BYT #FF,#FF,#FF,#FF
EBCC ....  BYT #FF,#FF,#FF,#FF
```

TABLE DE VECTEURS

```
EBDØ  JMP $ECC7  ..............  Autoriser IRQ
EBD3  JMP $ED8F  ..............  YX dans timer indexé par A
EBD6  JMP $ED81  ..............  lire dans YX timer No A
EBD9  JMP $EC8C  ..............  décalage mémoire
EBDC  JMP $EDAD  ..............  WAIT
EBDF  JMP $FØ2D  ..............  CURSET
EBE2  JMP $FØ64  ..............  CURMOV
EBE5  JMP $FØ79  ..............  DRAW
EBE8  JMP $F2E5  ..............  CIRCLE
EBEB  JMP $FØ93  ..............  PATTERN
EBEE  JMP $FØA5  ..............  CHAR
EBF1  JMP $F141  ..............  Evaluer couleur d'un point
EBF4  JMP $F17F  ..............  PAPER
EBF7  JMP $F18B  ..............  INK
```

```
EBFA  JMP $F1E5  ..............  FILL
EBFD  JMP $ED01  ..............  Interdire les IRQ
EC00  JMP $EDBC  ..............  effacer écran HIRES
EC03  JMP $ED09  ..............  IRQ
EC06  JMP $F264  ..............  comparaison 1
EC09  JMP $F250  ..............  comparaison 2
```

DECALER UN BLOC MEMOIRE

Entrée: #0200-#0201 contient le début du bloc à décaler (décalage vers le haut) ou la fin dans le cas contraire
#0202-#0203 contient la nouvelle adresse du bloc
#0204-#0205 contient la fin du bloc à décaler (décalage vers le haut) ou le début dans le cas contraire

Sortie: A,X,Y inchangés, C=1 si décalage vers le haut, C=0 sinon.

```
EC0C  PHA        ..............  sauver A
EC0D  TXA        ..............
EC0E  PHA        ..............  sauver X
EC0F  TYA        ..............
EC10  PHA        ..............  sauver Y
EC11  LDX #00    ..............  indexer #200
EC13  LDY #02    ..............  et #202
EC15  JSR $EC80  ..............  comparer (#200) et (#202)
EC18  BEQ EC7A   ..............  si égaux, on sort !
EC1A  PHP        ..............  sauver signe de la comparaison (dans C)
EC1B  LDY #00    ..............  préparer index
EC1D  BCS EC40   ..............  sauter si décalage vers le bas
EC1F  LDY #04    ..............  indexer #204
EC21  LDA #00    ..............
EC23  STA 0206   ..............
EC26  LDA #FF    ..............
EC28  STA 0207   ..............  #206-#207=#FF00=-#FF
EC2B  LDX #06    ..............  indexer #206
EC2D  LDY #04    ..............
EC2F  JSR $EC95  ..............  (#206)=haut du bloc-#FF
EC32  LDY #06    ..............
EC34  LDX #00    ..............
EC36  JSR $EC95  ..............  (#200)=bas du bloc+haut du bloc-#FF
EC39  LDX #02    ..............
EC3B  JSR $EC95  ..............  (#202)=adresse cible+haut du bloc-#FF
EC3E  LDY #FF    ..............  indiquer commencer par le haut
```

```
EC40  LDA #200  ..............
EC43  STA #C    ..............
EC45  LDA #201  ..............
EC48  STA #D    ..............  prendre adresse de départ
EC4A  LDA #202  ..............
EC4D  STA #E    ..............
EC4F  LDA #203  ..............
EC52  STA #F    ..............  et adresse cible
EC54  LDX #04   ..............  indexer la fin du bloc à décaler

Décaler

EC56  JSR $ECAF ..............  fin du bloc atteinte ?
EC59  BCC EC79  ..............  oui, sortir
EC5B  LDA (#C),Y ............
EC5D  STA (#E),Y ............  effectuer le transfert
EC5F  PLP       ..............  récupérer sens du décalage
EC60  PHP       ..............
EC61  BCS EC6F  ..............  sauter si vers le bas
EC63  DEY       ..............  décalage vers le haut, on décrémente donc
EC64  CPY #FF   ..............  index
EC66  BNE EC56  ..............
EC68  DEC #D    ..............
EC6A  DEC #F    ..............  et éventuellement poids fort
EC6C  JMP $EC56 ..............  (ou BCS) et continuer
EC6F  INY       ..............  décalage vers le bas, on incrémente
EC70  BNE EC56  ..............
EC72  INC #D    ..............
EC74  INC #F    ..............  et éventuellement poids fort
EC76  JMP $EC56 ..............  et on continue
EC79  PLP       ..............  ajuster la pile
EC7A  PLA       ..............
EC7B  TAY       ..............  récupérer Y
EC7C  PLA       ..............
EC7D  TAX       ..............  récuperer X
EC7E  PLA       ..............  récupérer A
EC7F  RTS       ..............
```

COMPARAISON

Entrée: X et Y indexent une position dans la page 2.

Sortie: C=1 si le nombre indexé par X est plus grand que celui indexé par Y

```
       Z=1 si ils sont égaux
       X et Y inchangés.

EC8Ø  CLD         ..............  pas de mode décimal (!)
EC81  SEC         ..............
EC82  LDA Ø2ØØ,X  ..............
EC85  SBC Ø2ØØ,Y  ..............  faire soustraction
EC88  STA Ø2Ø6    ..............  et sauver provisoirement
EC8B  LDA Ø2Ø1,X  ..............  idem poids fort
EC8E  SBC Ø2Ø1,Y  ..............
EC91  ORA Ø2Ø6    ..............  et positionner Z selon nombre nul ou non
EC94  RTS         ..............
```

FAIRE ADDITION

```
Entrée: X et Y indexent 2 nombres en page 2

Sortie: la somme des deux nombres est faite, le résultat est indexé par X
       C positionné comme pour toute addition
       Z=1 si le poids faible du résultat est nul.
       X et Y inchangés.

EC95  CLD         ..............
EC96  CLC         ..............
EC97  LDA Ø2ØØ,X  ..............
EC9A  ADC Ø2ØØ,Y  ..............
EC9D  STA Ø2ØØ,X  ..............  addition des poids faibles
ECAØ  PHA         ..............  sauver résultat sur la pile
ECA1  LDA Ø2Ø1,X  ..............
ECA4  ADC Ø2Ø1,Y  ..............
ECA7  STA Ø2Ø1,X  ..............  addition des poids forts
ECAA  PLA         ..............  récupérer poids faible
ECAB  ORA Ø2ØØ,X  ..............  inutile
ECAE  RTS         ..............
```

SOUSTRACTION

```
Entrée: X indexe un nombre en page 2

Sortie: le nombre est décrémenté, C est positionné comme pour une soustraction
       Z=1 si le poids faible du résultat est nul.
```

```
ECAF  CLD          .................
ECB0  SEC          .................
ECB1  LDA $0200,X  .................
ECB4  SBC #01      .................
ECB6  STA $0200,X  ................. décrémenter poids faible
ECB9  PHA          ................. sauver sur la pile aussi
ECBA  LDA $0201,X  .................
ECBD  SBC #00      .................
ECBF  STA $0201,X  ................. répercuter sur le poids fort
ECC2  PLA          ................. récupérer le poids faible
ECC3  ORA $0200,X  ................. inutile
ECC6  RTS          .................
```

DEPLACER UN BLOC VERS LE HAUT

Entrée: #0C-#0D contient le bas du bloc à décaler.
#0E-#0F contient l'adresse cible
#10-#11 le nombre d'octets du bloc.
Sortie: YX contient la longueur du bloc.

Attention: Cette routine n'effectue correctement que les décalages vers le haut, contrairement à la routine de la V1.0.

Vitesse: le temps moyen pour décaler un octet est de 20 microsecondes, contre 75 pour la V1.0, c'est à dire 4 fois plus rapide.
Pour le scroll d'un écran,(26*40=1040 caractéres), la V1.0 prend 80 ms environ, contre 21 ms à la V1.1
Une routine un peu plus optimisée permettrait de descendre à 14 microsecondes par octet,soit 15 ms par scroll.
NB: ce n'est pas la V1.1 qui est rapide mais la V1.0 qui est lente...

```
.................  EDC4  LDX #00     initialiser compteur de pages
.................  EDC6  LDY #00     et index
.................  EDC8  CPY 10
.................  EDCA  BNE EDD0
.................  EDCC  CPX 11
.................  EDCE  BEQ EDDF    test de fin
.................  EDD0  LDA (0C),Y  non,transfert
.................  EDD2  STA (0E),Y
.................  EDD4  INY         incrémenter index
.................  EDD5  BNE EDC8    et continuer
.................  EDD7  INC 0D      répercuter dans le poids fort
.................  EDD9  INC 0F
```

```
.............. EDDB  INX         et compteur aussi
.............. EDDC  JMP $EDC8   ...ou BNE
.............. EDDF  RTS
```

J) GESTION DES IRQ'S

1-Listing

AUTORISER IRQ PAR T1 (GESTION CLAVIER)

Remarque: cette routine s'occupe à la fois d'initialiser correctement le VIA, et de placer les timers soft qui régissent la gestion du clavier et du curseur.

Entrée: rien

Sortie: A inchangé

```
ECC7  PHA         EDE0  PHA         sauver A
ECC8  JSR $ED70   EDE1  JSR $EE8C   annuler les timers softs
ECCB  LDA #00     EDE4  LDA #00     indexer le timer 1
ECCD  LDX #00     EDE6  LDX #00     (gérer le clavier toutes les 3 IRQ)
ECCF  LDY #03     EDE8  LDY #03     et y mettre #0003
ECD1  JSR $ED8F   EDEA  JSR $EEAB
ECD4  LDA #01     EDED  LDA #01     indexer timer 2
ECD6  LDY #19     EDEF  LDY #19     (clignotement du curseur 1 fois sur 25 IRQ)
ECD8  JSR $ED8F   EDF1  JSR $EEAB
ECDB  LDA #00     EDF4  LDA #00
ECDD  STA 0271    EDF6  STA 0271    couleur du curseur: éteint
ECE0  LDA 030B    EDF9  LDA 030B    prendre VIA ACR
ECE3  AND #7F     EDFC  AND #7F     inhiber sortie sur PB7
ECE5  ORA #40     EDFE  ORA #40     et T1 en mode monostable
ECE7  STA 030B    EE00  STA 030B
ECEA  LDA #C0     EE03  LDA #C0     autoriser IRQ sur T1
ECEC  STA 030E    EE05  STA 030E
ECEF  LDA #10     EE08  LDA #10     charger timer avec #2710=10000
ECF1  STA 0306    EE0A  STA 0306    inutile, il suffit d'écrire dans le timer
ECF4  STA 0304    EE0D  STA 0304    timer poids faible (recopié dans le latche)
ECF7  LDA #27     EE10  LDA #27
ECF9  STA 0307    EE12  STA 0307    et poids fort dans le latche (démarrage)
ECFC  STA 0305    EE15  STA 0305    inutile, transfert lors du démarrage
ECFF  PLA         EE18  PLA         récupérer A
ED00  RTS         EE19  RTS
```

INHIBER IRQ CLAVIER

Entrée: rien

Sortie: A,X,Y inchangés

```
ED01  PHA          EE1A  PHA
ED02  LDA #40      EE1B  LDA #40      interdire IRQ par T1
ED04  STA 030E     EE1D  STA 030E     dans IER
ED07  PLA          EE20  PLA
ED08  RTS          EE21  RTS
```

IRQ

Entrée: par le vecteur #228/#244 normalement, P est au sommet de la pile

Sortie: aucun registre sauf P n'est affecté. Si l'IRQ n'est pas générée par T1, on sort tout de suite (une gestion du BRK eut été la bienvenue)

Il faut noter que la routine réautorise les IRQ, normalement inhibées automatiquement lors du recouvrement de l'IRQ par le 6502. Ce qui explique que l'Oric se plante si on met une valeur trop basse dans T1: une IRQ est générée pendant une autre IRQ, et la pile est petit à petit saturée.

```
ED09  PHA          EE22  PHA          sauver A
ED0A  LDA 030D     EE23  LDA 030D     prendre IFR
ED0D  AND #40      EE26  AND #40      et isoler b6, indicateur de T1
ED0F  BEQ ED17     EE28  BEQ EE30     si ce n'est pas lui, on sort
ED11  STA 030D     EE2A  STA 030D     enlever l'indicateur d'interruption
ED14  JSR $ED1B    EE2D  JSR $EE34    gestion de l'IRQ
ED17  PLA          EE30  PLA          récupérer A
ED18  JMP $0230    EE31  JMP $024A    et sortir (normalement RTI)
```

SOUS PROGRAMME IRQ

Action: s'occuppe de décrémenter les timers, de gérer le clavier une fois sur 3 et de changer la couleur du curseur (faire clignoter) une fois sur 25.

Entrée: Rien.

Sortie: A,X,Y inchangés, les IRQ sont réautorisées (CLI)

Bogue (corrigée):Sur la V1.0, les concepteurs avaient pensé que la #F43C

sortait avec X positionné comme il fallait, hélas il n'en est rien: un très laid rajout a été nécessaire. A croire qu'ils ne possédaient pas les sources de la ROM !

```
ED1B  PHA          EE34  PHA          sauver A
ED1C  TXA          EE35  TXA
ED1D  PHA          EE36  PHA          X
ED1E  TYA          EE37  TYA
ED1F  PHA          EE38  PHA          et Y
```

Gestion des timers soft

```
ED20  LDY #00      EE39  LDY #00      indexer le premier timer
ED22  LDA 0272,Y   EE3B  LDA 0272,Y
ED25  SEC          EE3E  SEC
ED26  SBC #01      EE3F  SBC #01      décrémenter le poids faible
ED28  STA 0272,Y   EE41  STA 0272,Y
ED2B  INY          EE44  INY
ED2C  LDA 0272,Y   EE45  LDA 0272,Y   et le poids fort éventuellement
ED2F  SBC #00      EE48  SBC #00
ED31  STA 0272,Y   EE4A  STA 0272,Y
ED34  INY          EE4D  INY
ED35  CPY #06      EE4E  CPY #06      a-t-on fait les trois timers ?
ED37  BNE ED22     EE50  BNE EE3B     non, on repart
```

Gerer le clavier

```
ED39  LDA #00      EE52  LDA #00      indexer timer 1
ED3B  JSR $ED81    EE54  JSR $EE9D    et lire dans YX sa valeur
ED3E  CPY #00      EE57  CPY #00      si pas 0,
ED40  BNE ED4F     EE59  BNE EE6B     sauter pour ne pas gérer le clavier
ED42  LDX #00      EE5B  LDX #00      si 0, on le recharge avec la valeur 3
ED44  LDY #03      EE5D  LDY #03
ED46  JSR $ED8F    EE5F  JSR $EEAB
.................  EE62  JSR $F495    gérer le clavier
.................  EE65  TXA          touche enfoncée ?
.................  EE66  BPL EE6B     non, on ne sauve pas
ED49  JMP $FC5E    .................  gérer le clavier
ED4C  STX 02DF     EE68  STX 02DF     sauver la touche
```

Clignotement curseur

```
ED4F  LDA #01      EE6B  LDA #01      indexer timer 2
ED51  JSR $ED81    EE6D  JSR $EE9D    et prendre sa valeur
```

```
ED54  CPY #00     EE70  CPY #00
ED56  BNE ED6A    EE72  BNE EE86    si pas 0, pas de clignotement
ED58  LDX #00     EE74  LDX #00     sinon,on le recharge à 25 (1/4 seconde)
ED5A  LDY #19     EE76  LDY #19
ED5C  JSR $ED8F   EE78  JSR $EEAB
ED5F  LDA 0271    EE7B  LDA 0271    on prend la couleur courante du curseur
ED62  EOR #01     EE7E  EOR #01     on l'inverse
ED64  STA 0271    EE80  STA 0271    on resauve
ED67  JSR $F403   EE83  JSR $F801   et on l'affiche si possible
ED6A  PLA         EE86  PLA
ED6B  TAY         EE87  TAY         récupérer Y
ED6C  PLA         EE88  PLA
ED6D  TAX         EE89  TAX         récupérer X
ED6E  PLA         EE8A  PLA         et enfin A...
ED6F  RTS         EE8B  RTS
```

ANNULER LES TROIS TIMERS SOFT

Entrée: rien

Sortie: les timers sont mis à zéro
A,X,Y inchangés

```
ED70  PHA         EE8C  PHA         sauver A
ED71  TYA         EE8D  TYA
ED72  PHA         EE8E  PHA         et Y
ED73  LDY #05     EE8F  LDY #05     indexer le 3ème timer
ED75  LDA #00     EE91  LDA #00
ED77  STA 0272,Y  EE93  STA 0272,Y  et vider...
ED7A  DEY         EE96  DEY         jusqu'à la fin
ED7B  BPL ED77    EE97  BPL EE93
ED7D  PLA         EE99  PLA
ED7E  TAY         EE9A  TAY         récupérer Y
ED7F  PLA         EE9B  PLA
ED80  RTS         EE9C  RTS         et A
```

PRENDRE VALEUR D'UN TIMER

Entrée: A contient le numéro du timer (0,1 ou 2)

Sortie: YX contient la valeur du timer
A est inchangé
Les IRQ sont réautorisées (CLI)

```
ED81  PHA          EE9D  PHA          sauver A
ED82  ASL A        EE9E  ASL A        calculer l'index
ED83  TAY          EE9F  TAY          dans Y
ED84  SEI          EEA0  SEI          éviter une décrémentation
ED85  LDA 0272,Y   EEA1  LDA 0272,Y   entre la lecture des poids faible et fort
ED88  LDX 0273,Y   EEA4  LDX 0273,Y   prendre valeur dans AX
ED8B  CLI          EEA7  CLI
ED8C  TAY          EEA8  TAY          puis dans YX
ED8D  PLA          EEA9  PLA          récupérer A
ED8E  RTS          EEAA  RTS
```

ECRIRE DANS UN TIMER

Entrée: A contient le numéro du timer (0 à 2)
YX contient la valeur à y placer
Sortie: A,X,Y inchangés
Les IRQ sont réautorisées (CLI)

```
ED8F  PHA          EEAB  PHA          sauver A
ED90  TXA          EEAC  TXA
ED91  PHA          EEAD  PHA          sauver X
ED92  TYA          EEAE  TYA
ED93  PHA          EEAF  PHA          sauver Y
ED94  TSX          EEB0  TSX
ED95  LDA 0103,X   EEB1  LDA 0103,X   récupérer valeur de A sur la pile
ED98  ASL A        EEB4  ASL A        calculer index
ED99  TAY          EEB5  TAY          dans Y
ED9A  PLA          EEB6  PLA          récupérer Y sur la pile (poids faible)
ED9B  PHA          EEB7  PHA          et ajuster la pile
ED9C  SEI          EEB8  SEI          inhiber IRQ
ED9D  STA 0272,Y   EEB9  STA 0272,Y   et sauver poids faible
EDA0  LDA 0102,X   EEBC  LDA 0102,X   récupérer X sur la pile (poids fort)
EDA3  STA 0273,Y   EEBF  STA 0273,Y   et sauver poids fort
EDA6  CLI          EEC2  CLI          réautoriser IRQ
EDA7  PLA          EEC3  PLA
EDA8  TAY          EEC4  TAY          récupérer Y
EDA9  PLA          EEC5  PLA
EDAA  TAX          EEC6  TAX          récupérer X
EDAB  PLA          EEC7  PLA          et enfin A
EDAC  RTS          EEC8  RTS
```

ATTENDRE QU'UN TIMER S'ANNULE

Entrée: A contient le numéro du timer
YX contient la valeur à y placer

Sortie: le timer considéré s'est annulé (si les IRQ étaient autorisées)
A est inchangé
X=Y=Ø,Z=1,C=1

```
EDAD  JSR $ED8F    EEC9  JSR $EEAB  écrire dans le timer la valeur voulue
EDBØ  JSR $ED81    EECC  JSR $EE9D  prendre la valeur du timer
EDB3  CPY #ØØ      EECF  CPY #ØØ    tester poids faible
EDB5  BNE EDBØ     EED1  BNE EECC   si pas nul, on attend
EDB7  CPX #ØØ      EED3  CPX #ØØ    si nul, tester poids fort aussi
EDB9  BNE EDBØ     EED5  BNE EECC   si non nul, on attend...
EDBB  RTS          EED7  RTS
```

K) ROUTINES GRAPHIQUES

1-Généralités

Les routines graphiques sont très faciles à utiliser, aussi bien en BASIC qu'en assembleur.

Toutes ces routines s'articulent autour de la gestion du 'curseur', c'est à dire le point courant. Le curseur est défini par quatre paramètres, qui sont tous intimement liés, et qu'il ne faut pas modifier indépendamment les uns des autres sous peines de surprises.

-La coordonnée horizontale (#219), de Ø à 239

-La coordonnée verticale (#21A), de Ø à 199

-L'adresse à l'écran du curseur (adresse de l'octet qui contient le curseur). C'est le pointeur #1Ø-#11

-Le masque du point dans l'octet (appelé sextet dans le texte), c'est à dire un un dans un bit (bØ à b5) qui indique sa position.

A partir de ces paramètres, il existe des routines pour déplacer le 'curseur' dans les quatre directions, en ajustant uniquement l'adresse et le masque (#1ØØ-#11 et #215).

Une routine graphique, pour être compatible avec les routines existantes, doit prendre garde à la cohérence des quatres paramètres cités plus haut, quitte à sauver (comme le CIRCLE) l'adresse et le masque pour les récupérer, si la routine y touche.

Les algorithmes de tracé de droites et de cercles seront décrits dans le listing. Attachons nous aux principes généraux.

L'écran haute résolution occuppe 8000 octets de #A000 à #BF3F, 40 octets formant une ligne, et il y a bien sur 200 lignes.

Chaque octet permet d'afficher 6 points, ou un attribut vidéo (couleur, clignotement etc...). S'il doit afficher des points, b0 à b5 représentent le motif binaire (1: couleur d'encre, 0: couleur de fond), b6 est à 0 et b7 indique l'éventuelle inversion des couleurs. Pour plus de précisions, voir l'annexe 'Codes d'affichage'.

Pour trouver l'adresse d'un point de coordonnées X,Y, il faut calculer AD=#A000+Y*40+INT(X/6) et il reste donc un nombre de 0 à 5 qui représente la position du bit dans l'octet (le sextet) courant. On comprend que des commandes du type CURSET soient lentes, puisque nécessitant une division et une multiplication.

Si on veut agir rapidement, il est préférable d'agir relativement au point courant: pour passer à la ligne précédente, il suffit d'ajouter 40 à l'adresse, pour passer au point à droite, il suffit d'une rotation à droite, suivi éventuellemnt d'un report sur l'octet suivant.

En résumé, il est conseillé d'utiliser au minimum les routines de type CURSET, pour essayer de travailler 'en relatif'. Et, aussi de veiller à la cohérence des variables #10-#11-#219-#21A-#215.

2-Listing

EFFACER ECRAN HIRES

Entrée: rien de spécial

Sortie: A,X et Y inchangés
L'écran est rempli de #40 (effacé)

```
EDBC  PHA      ..............  sauver A
EDBD  TXA      ..............
EDBE  PHA      ..............  sauver X
EDBF  TYA      ..............
EDC0  PHA      ..............  sauver Y (utilité ?)
EDC1  LDA #00  ..............
EDC3  STA 0C   ..............
EDC5  LDA #BF  ..............
EDC7  STA 0D   ..............  initialiser pointeur sur #BF00
EDC9  LDX #1F  ..............  il y a 32 pages à effacer
EDCB  LDA #40  ..............  prendre code écran noir
```

```
EDCD  LDY #3F     ..............  et commencer en #BF3F
EDCF  STA (ØC),Y  ..............  envoyer à l'écran
EDD1  DEY         ..............
EDD2  CPY #FF     ..............  fin de la page atteinte ?
EDD4  BNE EDCF    ..............  non, on repart
EDD6  DEC ØD      ..............  oui, ajuster poids fort sur page précédente
EDD8  DEX         ..............  et décrémenter le compteur
EDD9  CPX #FF     ..............  et tester si fin
EDDB  BNE EDCF    ..............  (DEX/BPL EDCF aurait été mieux !)
EDDD  PLA         ..............
EDDE  TAY         ..............
EDDF  PLA         ..............
EDEØ  TAX         ..............
EDE1  PLA         ..............  et récupérer les registres
EDE2  RTS         ..............
```

PLACER FB CODE

Entrée: le FB code est en #212 dans bØ-b1
Sortie: il est déplacé vers b6-b7 pour faciliter le test par N et V
X et Y inchangés
V1.1: sauve aussi le registre de pattern en #214.

```
EDE3  ASL Ø212    ..............
EDE6  ASL Ø212    ..............  décaler FB code de bØ-b1 vers b6-b7
EDE9  ASL Ø212    ..............
EDEC  ASL Ø212    ..............
EDEF  ASL Ø212    ..............  avec six rotations à gauche
EDF2  ASL Ø212    ..............
EDF5  RTS         ..............
..............  EED8  LDA Ø213   prendre registre pattern
..............  EEDB  STA Ø214   et le sauver
..............  EEDE  LSR Ø212
..............  EEE1  ROR Ø212   décaler FB code de bØ-b1 vers b6-b7
..............  EEE4  ROR Ø212   avec 3 rotations à droite via C
..............  EEE7  RTS
```

AFFICHER UN POINT (coordonnées X,Y)

Entrée: FB code en #212 (bØ-b1)
X,Y contiennent les coordonnées du point.

Sortie: le point est affiché (selon FB code), #215 contient le sextet le concernant, et #10-#11 l'adresse de la ligne.
A et Y inchangés.

Remarque: cette routine est assez lente car elle impose une division par 40.

```
EDF6  PHA        EEE8  PHA        sauver A
EDF7  TYA        EEE9  TYA
EDF8  PHA        EEEA  PHA        sauver Y
EDF9  JSR $EDE3  EEEB  JSR $EEDE  ajuster FB code
EDFC  JSR $EFA6  EEEE  JSR $F049  calculer motif du point
EDFF  JSR $EF5B  EEF1  JSR $F024  et afficher selon motif
EE02  PLA        EEF4  PLA
EE03  TAY        EEF5  TAY        récupérer Y
EE04  PLA        EEF6  PLA        récupérer A
EE05  RTS        EEF7  RTS
```

DRAW (CALCUL ET AFFICHAGE)

Entrée: #212:FB code

V1.0:#202-#203:DX
#204-#204:DY

V1.1:#2E1-#2E2:DX
#2E3-#2E4:DY

Sortie: A,X,Y inchangés sur V1.0.

Principe: on va tracer en décrivant l'axe pour lequel le déplacement est le plus grand. On pourra ainsi passer d'un point à un autre seulement en incrémentant une des coordonnées, et en calculant l'autre.
C'est le seul moyen pour tracer des droites verticales (la tangente serait infinie), et donc en fait de se ramener à une fonction.
Pour calculer le pas (la tangente en fait), il faut la valeur absolue de DX ou DY. Mais comme on ne se sert que du poids faible, le poids fort garde le signe du déplacement.

```
EE06  PHA        ..............  sauver A
EE07  TXA        ..............
EE08  PHA        ..............  sauvr X
EE09  TYA        ..............
EE0A  PHA        ..............  sauver Y
```

```
Ajuster les paramètres

EE0B  LDA 0213    ..............
EE0E  STA 0214    ..............  sauver le registre PATTERN
EE11  JSR $EDE3   EEF9  JSR $EED8  ajuster FB code
EE14  BIT 0203    EEFC  BIT 02E2   DX positif ?
EE17  BPL EE24    EEFF  BPL EF0B   oui, rien à faire
EE19  LDA #FF     EF01  LDA #FF    non, on complémente le poids faible
EE1B  EOR 0202    EF03  EOR 02E1   car c'est sa valeur absolue qu'on veut
EE1E  STA 0202    ..............
EE21  INC 0202    ..............
..............    EF06  TAX
..............    EF07  INX
..............    EF08  STX 02E1
EE24  BIT 0205    EF0B  BIT 02E4   DY positif ?
EE27  BPL EE34    EF0E  BPL EF1A   oui, rien á faire
EE29  LDA #FF     EF10  LDA #FF    non, on complémente le poids faible
EE2B  EOR 0204    EF12  EOR 02E3
EE2E  STA 0204    ..............
EE31  INC 0204    ..............
..............    EE15  STA 0305
EE34  LDA 0202    EF1A  LDA 02E1   qui de DX ou DY est le plus grand ?
EE37  CMP 0204    EF1D  CMP 02E3
EE3A  BCC EE66    EF20  BCC EF31   c'est DY, on saute

Tracer en décrivant l'axe des X

EE3C  LDA #00     ..............
EE3E  STA 0C      ..............
EE40  STA 0201    ..............
EE43  LDA 0204    ..............  #0C-#0D=DY (##100)
EE46  STA 0D      ..............
EE48  LDA 0202    ..............
EE4B  STA 0200    ..............  et #200-#201=DX
EE4E  JSR $EEFF   ..............  calculer DY/DX (##100)
EE51  JSR $EF31   ..............  arrondir le quotient (la tangente)
EE54  LDA #00     ..............
EE56  STA 0E      ..............
EE58  STA 0F      ..............  valeur de départ=0
EE5A  STA 0200    ..............
EE5D  LDX 0202    ..............  prendre dans X le nombre de point à tracer
EE60  JSR $EEBB   ..............  et tracer la droite
EE63  JMP $EE8D   ..............  et finir...
..............    EF22  LDX 02E1   prendre DX
```

```
.............. EF25  BEQ EF3Ø    nul, on sort
.............. EF27  LDA Ø2E3    prendre DY
.............. EF2A  JSR $EF4Ø   calculer DY/DX (**1ØØ)
.............. EF2D  JSR $EF84   et tracer la droite
.............. EF3Ø  RTS
```

Tracer en décrivant l'axe des Y

```
EE66  LDA #ØØ     ..............
EE68  STA ØC      ..............
EE6A  STA Ø2Ø1    ..............
EE6D  LDA Ø2Ø2    ..............
EE7Ø  STA ØD      .............. #ØC-#ØD=DX (**1ØØ)
EE72  LDA Ø2Ø4    ..............
EE75  STA Ø2ØØ    .............. #2ØØ-#2Ø1=DY
.............. EF31  LDX Ø2E3    si DY nul, on sort
.............. EF34  BEQ EF3F
.............. EF36  LDA Ø2E1    prendre DX
.............. EF39  JSR $EF4Ø   et calculer DX/DY (**1ØØ)
.............. EF3C  JSR $EF5C   et tracer la droite
.............. EF3F  RTS

.............. EF4Ø  STA ØD      sauver DX ou DY
.............. EF42  STX Ø2ØØ    sauver DX ou DY
.............. EF45  LDA #ØØ
.............. EF47  STA ØC      annuler poids faibles
.............. EF49  STA Ø2Ø1    ou fort
EE78  JSR $EEFF   EF4C  JSR $EFC8   calculer DX/DY (**1ØØ) (ou DY/DX pour V1.1)
EE7B  JSR $EF31   EF4F  JSR $EFFA   arrondir le quotient (la tangente)
EE7E  LDA #ØØ     EF52  LDA #ØØ
EE8Ø  STA ØE      EF54  STA ØE
EE82  STA ØF      EF56  STA ØF      valeur de départ=Ø
EE84  STA Ø2ØØ    EF58  STA Ø2ØØ
EE87  LDX Ø2Ø4    .............. prendre le nombre de points à tracer
EE8A  JSR $EE93   .............. et tracer la droite
EE8D  PLA         ..............
EE8E  TAY         .............. récupérer Y
EE8F  PLA         ..............
EE9Ø  TAX         .............. récupérer X
EE91  PLA         .............. récupérer A
EE92  RTS         EF5B  RTS
```

TRACER UNE DROITE SELON L'AXE DES Y

```
EE93  BIT Ø2Ø5    EF5C  BIT Ø2E4    prendre signe de DY
EE96  BPL EE9E    EF5F  BPL EF67    positif,on saute
EE98  JSR $EFF5   EF61  JSR $F095   négatif, déplacer vers le haut
EE9B  JMP $EEA1   EF64  JMP $EF6A   et test de fin
EE9E  JSR $EFE6   EF67  JSR $F089   déplacer verticalement vers le bas
EEA1  JSR $EEE3   EF6A  JSR $EFAC   calculer le point suivant
EEA4  BEQ EEB4    EF6D  BEQ EF7D    c'est le même, test de fin
EEA6  BIT Ø2Ø3    EF6F  BIT Ø2E2    sinon, tester déplacement selon X
EEA9  BPL EEB1    EF72  BPL EF7A    sauter si positif
EEAB  JSR $F015   EF74  JSR $F0B2   déplacer horizontalement à gauche
EEAE  JMP $EEB4   EF77  JMP $EF7D   et suite
EEB1  JSR $F004   EF7A  JSR $F0A1   déplacer horizontalement à droite
EEB4  JSR $EF4D   EF7D  JSR $F016
EEB7  DEX         EF80  DEX         et décompter le nombre de positions
EEB8  BNE EE93    EF81  BNE EF5C    recommencer si pas fini
EEBA  RTS         EF83  RTS
```

TRACER UNE DROITE SELON L'AXE DES X

```
EEBB  BIT Ø2Ø3    EF84  BIT Ø2E2    tester signe de DX
EEBE  BPL EEC6    EF87  BPL EF8F    positif, sauter
EEC0  JSR $F015   EF89  JSR $F0B2   négatif, déplacer un point à gauche
EEC3  JMP $EEC9   EF8C  JMP $EF92   et suite
EEC6  JSR $F004   EF8F  JSR $F0A1   déplacer un point à droite
EEC9  JSR $EEE3   EF92  JSR $EFAC   calculer Y+DY
EECC  BEQ EEDC    EF95  BEQ EFA5    si pas changé, sauter
EECE  BIT Ø2Ø5    EF97  BIT Ø2E4    tester signe de DY
EED1  BPL EED9    EF9A  BPL EFA2    positif, on saute
EED3  JSR $EFF5   EF9C  JSR $F095   négatif, déplacer vers le haut
EED6  JMP $EEDC   EF9F  JMP $EFA5   et suite
EED9  JSR $EFE6   EFA2  JSR $F089   déplacer vers le bas
EEDC  JSR $EF4D   EFA5  JSR $F016   afficher le point...
EEDF  DEX         EFA8  DEX         et décompter le nombre de points
EEE0  BNE EEBB    EFA9  BNE EF84
EEE2  RTS         EFAB  RTS
```

DRAW:CALCULER LE POINT SUIVANT

Entrée: $0C-$0D contient l'incrément (non signé), i.e la tangente
$0E-$0F contient la position courante

Sortie: #ØE-#ØF contient sa nouvelle valeur
#2ØØ et A contiennent le poids fort du résultat, éventuellement arrondi (si le poids fort est supérieur à 128)
Z=Ø si le contenu de #2ØØ a changé, c'est à dire si le poids fort a été changé.

```
EEE3  CLD          EFAC  CLD          interdire le mode décimal
EEE4  CLC          EFAD  CLC
EEE5  LDA ØE       EFAE  LDA ØE
EEE7  ADC ØC       EFBØ  ADC ØC
EEE9  STA ØE       EFB2  STA ØE       ajouter valeur courante et pas
EEEB  LDA ØF       EFB4  LDA ØF
EEED  ADC ØD       EFB6  ADC ØD
EEEF  STA ØF       EFB8  STA ØF       poids fort aussi
EEF1  BIT ØE       EFBA  BIT ØE       tester poids faible
EEF3  BPL EEF8     EFBC  BPL EFC1     si <128,pas d'arrondi
EEF5  CLC          EFBE  CLC          sinon,arrondir (+1
EEF6  ADC #Ø1      EFBF  ADC #Ø1
EEF8  CMP Ø2ØØ     EFC1  CMP Ø2ØØ     placer Z selon ancienne valeur
EEFB  STA Ø2ØØ     EFC4  STA Ø2ØØ     et sauver nouvelle valeur.
EEFE  RTS          EFC7  RTS
```

DIVISION ENTIERE

Entrée: #ØC-#ØD et #2ØØ-#2Ø1 contiennent deux nombres non signés.

Sortie: #ØC-#ØD contient le quotient (#ØC-#ØD/#2ØØ-#2Ø1) et #ØE-#ØF le reste.
A,X et Y inchangés.

Principe: classique, mais formellement un peu compliqué car #ØC-#ØD sert à la fois de dividende original et de quotient .
Le principe est exactement identique à une division en décimal, avec la simplification qu'il n'y a que deux possibilités: soit le diviseur est inférieur au reste courant, soit il est plus grand, et on fait la soustraction. En décimal, il y a chaque fois dix chiffres à essayer.

```
EEFF  PHA          EFC8  PHA
EFØØ  TXA          EFC9  TXA
EFØ1  PHA          EFCA  PHA
EFØ2  TYA          EFCB  TYA
EFØ3  PHA          EFCC  PHA          sauver les registres
EFØ4  LDA #ØØ      EFCD  LDA #ØØ
EFØ6  STA ØE       EFCF  STA ØE       reste (ou dividende courant) à Ø
```

```
EF08  STA 0F      EFD1  STA 0F
EF0A  LDX #10     EFD3  LDX #10     le diviseur a 16 bits
EF0C  ASL 0C      EFD5  ASL 0C      quotient multiplié par 2,et sortir
EF0E  ROL 0D      EFD7  ROL 0D      un bit du dividende original
EF10  ROL 0E      EFD9  ROL 0E      ajuster nouveau dividende (reste courant)
EF12  ROL 0F      EFDB  ROL 0F
EF14  LDA 0E      EFDD  LDA 0E
EF16  SEC         EFDF  SEC         faire dividende moins diviseur
EF17  SBC 0200    EFE0  SBC 0200
EF1A  TAY         EFE3  TAY
EF1B  LDA 0F      EFE4  LDA 0F
EF1D  SBC 0201    EFE6  SBC 0201    et résultat dans YA
EF20  BCC EF28    EFE9  BCC EFF1    si dividende trop petit, suivant
EF22  INC 0C      EFEB  INC 0C      sinon, ajuster résultat (b0=1)
EF24  STY 0E      EFED  STY 0E
EF26  STA 0F      EFEF  STA 0F      et nouveau reste  (dividende courant)
EF28  DEX         EFF1  DEX         16 bits passés ?
EF29  BNE EF0C    EFF2  BNE EFD5    non, recommencer
EF2B  PLA         EFF4  PLA
EF2C  TAY         EFF5  TAY
EF2D  PLA         EFF6  PLA
EF2E  TAX         EFF7  TAX
EF2F  PLA         EFF8  PLA         récupérer les registres
EF30  RTS         EFF9  RTS
```

ARRONDIR QUOTIENT (?)

Entrée: #0C-#0D,#0E-#0F,#200-#201 placés par la routine de division ci-dessus.

Sortie: #0C-#0D (quotient) arrondi d'après le reste et le diviseur.
A,X et Y inchangés.

Bogue: (?) par définition, le reste est inférieur au diviseur, et à fortiori au double du diviseur. Cette routine ne fera donc jamais rien.
Il aurait en fait fallu comparer le reste à la moitié du diviseur, ou le double du reste au diviseur, pour arrondir correctement. Curieux !

```
EF31  PHA         EFFA  PHA         sauver A
EF32  ASL 0200    EFFB  ASL 0200
EF35  ROL 0201    EFFE  ROL 0201    calculer le double du diviseur
EF38  LDA 0200    F001  LDA 0200
EF3B  SEC         F004  SEC
EF3C  SBC 0E      F005  SBC 0E      et lui retrancher le reste,
```

```
EF3E  LDA #201    F007  LDA #201
EF41  SBC #F      F00A  SBC #F
EF43  BCS EF4B    F00C  BCS F014    qui lui est toujours inférieur...
EF45  INC #C      F00E  INC #C      et on n'arrondi donc jamais le quotient.
EF47  BNE EF4B    F010  BNE F014
EF49  INC #D      F012  INC #D      tout celà est-il bien normal ?
EF4B  PLA         F014  PLA         récupérer A
EF4C  RTS         F015  RTS
```

AFFICHER UN POINT

Entrée: #212 contient le FB code
#214 contient le motif courant du Pattern
#215 contient le masque du point à afficher.

Sortie: #214 ajusté
X inchangé.

```
EF4D  BIT #214    F016  BIT #214    tester bit sortant du pattern courant
EF50  CLC         F019  CLC         préparer pour réentrer un 0
EF51  BPL EF57    F01A  BPL F020    si 0,pas d'affichage,simplement rotation
EF53  JSR $EF5B   F01C  JSR $F024   afficher le point
EF56  SEC         F01F  SEC         il va rentrer un 1
EF57  ROL #214    F020  ROL #214    et décaler le pattern courant,cycliquement
EF5A  RTS         F023  RTS

.....             F024  LDY #00     préparer index
.....             F026  LDA (10),Y  prendre sextet courant
.....             F028  AND #40     tester si attribut (b6=0)
.....             F02A  BEQ $F048   oui,on sort
.....             F02C  LDA #215    prendre motif point
EF5B  BIT #212    F02F  BIT #212    tester FB code
EF5E  BMI EF6E    F032  BMI $F042   sauter si 2 ou 3
EF60  BVS EF68    F034  BVS $F03D   sauter si 1
```

FB code=0

```
EF62  JSR $EF94   .....             traiter FB code=0
.....             F036  EOR #FF     inverser le masque
.....             F038  AND (10),Y  forcer point éteint
.....             F03A  STA (10),Y  et sauver
EF65  JMP $EF73   F03C  RTS         bien joué sur la V1.0
```

```
FB code=1

EF68  JSR $EF84   ..............  traiter FB code=1
..............  F03D  ORA (10),Y  forcer le point allumé
..............  F03F  STA (10),Y  et sauver
EF6B  JMP $EF73   F041  RTS
EF6E  BVS EF73    F042  BVS $F048  c'est tout si FB code=3

FB code =2


EF70  JSR $EF74   ..............  traiter FB code=2
..............  F044  EOR (10),Y  changer la couleur du point
..............  F046  STA (10),Y  et sauver

FB code =3 (dur !)

EF73  RTS         F048  RTS

EF74  LDY #02     ..............
EF76  LDA (10),Y  ..............  prendre sextet courant
EF78  AND #40     ..............  et tester si attribut (b4=0)
EF7A  BEQ EF83    ..............  oui,on sort
EF7C  LDA (10),Y  ..............  prendre sextet courant
EF7E  EOR 0215    ..............  inverser le bit concerné
EF81  STA (10),Y  ..............  et replacer
EF83  RTS         ..............

EF84  LDY #00     ..............
EF86  LDA (10),Y  ..............
EF88  AND #40     ..............
EF8A  BEQ EF93    ..............  sortir si attribut vidéo
EF8C  LDA 0215    ..............  prendre motif
EF8F  ORA (10),Y  ..............  et forcer le point
EF91  STA (10),Y  ..............  et resauver
EF93  RTS         ..............

EF94  LDY #00     ..............
EF96  LDA (10),Y  ..............
EF98  AND #40     ..............
EF9A  BEQ EFA5    ..............  sortir si attribut vidéo
EF9C  LDA 0215    ..............  prendre motif
EF9F  EOR #FF     ..............  et inverser le masque
EFA1  AND (10),Y  ..............  forcer le point à 0
```

```
EFA3  STA (1Ø),Y  ..............  et écrire
EFA5  RTS         ..............
```

CALCULER MOTIF ET ADRESSE DU POINT

Entrée: X et Y contiennent les coordonnées horizontale et verticale du point désiré

Sortie: #1Ø-#11 contient l'adresse du sextet qui comprend le point
#Ø215 contient le masque du point dans le septet
A et Y inchangés.

```
EFA6  CLD          F049  CLD          pas décimal (une fois de plus !)
EFA7  PHA          F04A  PHA          sauver A
EFA8  TYA          F04B  TYA
EFA9  PHA          F04C  PHA          sauver Y
EFAA  JSR $F4ØØ    F04D  JSR $F731    calculer 4Ø*A
EFAD  CLC          F05Ø  CLC
EFAE  ADC #ØØ      F051  ADC #ØØ      très utile ! (à cause d'une étiquette ??)
EFBØ  STA 1Ø       F053  STA 1Ø       ajouter #AØØØ
EFB2  TYA          F055  TYA          et #Ø-#11 contient l'adresse du début
EFB3  ADC #AØ      F056  ADC #AØ      de la ligne demandée
EFB5  STA 11       F058  STA 11
EFB7  LDA #ØØ      F05A  LDA #ØØ
EFB9  STA ØD       F05C  STA ØD
EFBB  STA Ø2Ø1     F05E  STA Ø2Ø1
EFBE  STX ØC       F061  STX ØC       #ØC-#ØD=X
EFCØ  LDA #Ø6      F063  LDA #Ø6
EFC2  STA Ø2ØØ     F065  STA Ø2ØØ     #2ØØ-#2Ø1=6
EFC5  JSR $EEFF    F068  JSR $EFC8    calculer X/6,soit la position dans la ligne
EFC8  CLC          F06B  CLC
EFC9  LDA ØC       F06C  LDA ØC       prendre la position
EFCB  ADC 1Ø       F06E  ADC 1Ø       ajouter à adresse de la ligne
EFCD  STA 1Ø       F07Ø  STA 1Ø       on obtient l'adresse du point
EFCF  LDA #ØØ      F072  LDA #ØØ      répercuter dans le poids fort
EFD1  ADC 11       F074  ADC 11       (BCC/INC 11 aurait eu plus d'effet !)
EFD3  STA 11       F076  STA 11
EFD5  LDA #2Ø      F078  LDA #2Ø      préparer motif pour point à droite
EFD7  LDY ØE       F07A  LDY ØE       prendre le reste (Ø à 5)
EFD9  BEQ EFDF     F07C  BEQ F082     si nul,la position est la bonne,on sort
EFDB  LSR A        F07E  LSR A        motif point à droite
EFDC  DEY          F07F  DEY          reste-1
EFDD  BCC EFD9     F08Ø  BCC F07C     inconditionnel:on continue...
```

```
EFDF  STA Ø215    F082  STA Ø215    sauver motif du point
EFE2  PLA         F085  PLA
EFE3  TAY         F086  TAY         récupérer Y
EFE4  PLA         F087  PLA         récupérer A
EFE5  RTS         F088  RTS
```

DEPLACER VERS LE BAS

Action: calcule l'adresse et le masque du point au dessous du point courant.

```
EFE6  CLD         ..............  pas de mode décimal
EFE7  CLC         F089  CLC
EFE8  LDA 1Ø      F08A  LDA 1Ø
EFEA  ADC #28     F08C  ADC #28
EFEC  STA 1Ø      F08E  STA 1Ø      adresse du sextet + 4Ø (ligne suivante)
EFEE  LDA 11      F090  BCC F094
EFFØ  ADC #ØØ     F092  INC 11
EFF2  STA 11      ..............
EFF4  RTS         F094  RTS
```

DEPLACER VERS LE HAUT

Action: calcule l'adresse et le masque du point au dessus du point courant.

```
EFF5  CLD         ..............
EFF6  SEC         F095  SEC
EFF7  LDA 1Ø      F096  LDA 1Ø
EFF9  SBC #28     F098  SBC #28
EFFB  STA 1Ø      F09A  STA 1Ø      adresse du sextet -4Ø (ligne précédente)
EFFD  LDA 11      F09C  BCS F0A0
EFFF  SBC #ØØ     F09E  DEC 11
F001  STA 11      ..............
F003  RTS         F0A0  RTS
```

DEPLACER VERS LA DROITE

Action: calcule l'adresse et le masque du point à droite du point courant.

```
F004  LSR Ø215    F0A1  LSR Ø215    décaler le motif à droite
F007  BCC F014    F0A4  BCC F0B1    si le point ne 'sort' pas,c'est tout
```

```
F009  LDA #20     F0A6  LDA #20
F00B  STA 0215    F0A8  STA 0215    sinon,motif=point le plus à gauche
F00E  INC 10      F0AB  INC 10      mais du sextet suivant
F010  BNE F014    F0AD  BNE F0B1
F012  INC 11      F0AF  INC 11
F014  RTS         F0B1  RTS
```

DEPLACER VERS LA GAUCHE

Action: calcule l'adresse et le masque à gauche du point courant.

```
F015  ASL 0215    F0B2  ASL 0215    décaler le motif à gauche
F018  BIT 0215    F0B5  BIT 0215    et tester si le point 'sort' à gauche
F01B  BVC F02C    F0B8  BVC F0C7    non, on sort
F01D  LDA #01     F0BA  LDA #01
F01F  STA 0215    F0BC  STA 0215    oui, motif =point le plus à droite
F022  DEC 10      ..............
F024  LDA 10      ..............
F026  CMP #FF     ..............
F028  BNE F02C    ..............    mais du sextet précédent
F02A  DEC 11      ..............
..............    F0BF  LDA 10
..............    F0C1  BNE F0C5
..............    F0C3  DEC 11
..............    F0C5  DEC 10
F02C  RTS         F0C7  RTS
```

'CURSET' (COMMANDE)

Entrée: #2E1-#2E2 contient la position horizontale
#2E3-#2E4 contient la position verticale
#2E5-#2E6 contient le FB code

Sortie: #2E0 incrémenté si erreur de paramètre.

Remarque: assez lent car nécessitant une division.

```
F02D  LDA #04     F0C8  LDA #04
F02F  LDX #E5     F0CA  LDX #E5     indexer #2E5
F031  JSR $F264   F0CC  JSR $F2F8   FB code plus petit que 4 ?
F034  BCS F060    F0CF  BCS F0F9    non,erreur
F036  LDA 02E5    F0D1  LDA 02E5
```

```
F039  STA 0212    F0D4  STA 0212    oui,sauver FB code
F03C  LDA #F0     F0D7  LDA #F0     240
F03E  LDX #E1     F0D9  LDX #E1     indexer #2E1
F040  JSR $F264   F0DB  JSR $F2F8   coordonnée horizontale plus petite que 240
F043  BCS F060    F0DE  BCS F0F9    non,erreur
F045  LDA #C8     F0E0  LDA #C8     200
F047  LDX #E3     F0E2  LDX #E3     indexer #2E3
F049  JSR $F264   F0E4  JSR $F2F8   coordonée verticale plus petite que 200 ?
F04C  BCS F060    F0E7  BCS F0F9    non,erreur
F04E  LDX 02E1    F0E9  LDX 02E1
F051  STX 0219    F0EC  STX 0219    placer nouvelle coordonnée horizontale
F054  LDY 02E3    F0EF  LDY 02E3
F057  STY 021A    F0F2  STY 021A    et verticale
F05A  JSR $EDF6   F0F5  JSR $EEE8   calculer adresse et motif du curseur
F05D  JMP $F063   F0F8  RTS
F060  INC 02E0    F0F9  INC 02E0    si erreur,incrémenter #2E0 (1 normalement)
F063  RTS         F0FC  RTS
```

'CURMOV' (COMMANDE)

Entrée: #2E1-#2E2 contient le déplacement horizontal (signé)
#2E3-#2E4 contient le déplacement vertical (signé)
#2E5-#2E6contient le FB code

Sortie: #2E0 incrémenté si erreur dans les paramètres.

```
F064  JSR $F276   F0FD  JSR $F30A   vérifier les paramètres relatifs
F067  BCS F075    F100  BCS F10C    et sortir si erreur
F069  LDX 0219    F102  LDX 0219
F06C  LDY 021A    F105  LDY 021A    prendre nouvelles coordonnées
F06F  JSR $EDF6   F108  JSR $EEE8   et calculer adresse et motif
F072  JMP $F078   F10B  RTS
F075  INC 02E0    F10C  INC 02E0
F078  RTS         F10F  RTS
```

'DRAW' (COMMANDE)

Entrée: #2E1-#2E2 contient le déplacement horizontal (signé)
#2E3-#2E4 contient le déplacement vertical (signé)
#2E5-#2E6 contient le FB code.

Sortie: #2E0 incrémenté si paramètres incorrects.

```
F079  JSR $F276   F110  JSR $F30A   vérifier les paramètres relatifs
F07C  BCS F08F    F113  BCS F119    et sortir si erreur
F07E  LDX #04     ..................
F080  LDA 02E0,X  ..................
F083  STA 0201,X  ..................
F086  DEX         ..................  transfert des paramètres dans zone travail
F087  BNE F080    ..................
F089  JSR $EE06   F115  JSR $EEF8    calcul et tracé de la droite
F08C  JMP $F092   F118  RTS
F08F  INC 02E0    F119  INC 02E0
F092  RTS         F11C  RTS
```

'PATTERN' (COMMANDE)

Entrée: #2E1-#2E2 contient le motif.

Sortie: #2E0 incrémenté si paramètre trop grand.

```
F093  LDX 02E2    F11D  LDX 02E2    paramètre négatif ou trop grand ?
F096  BNE F0A1    F120  BNE F129    oui, erreur
F098  LDX 02E1    F122  LDX 02E1    prendre argument
F09B  STX 0213    F125  STX 0213    comme registre de pattern
F09E  JMP $F0A4   F128  RTS
F0A1  INC 02E0    F129  INC 02E0
F0A4  RTS         F12C  RTS
```

'CHAR' (COMMANDE)

Entrée: #2E1-#2E2 contient le code ASCII du caractère à afficher
#2E3-#2E4 contient le numéro du jeu de caractère.
#2E5-#2E6 contient le FB code.

Sortie: #2E0 incrémenté si paramètres incorrects.

```
F0A5  LDX 02E2    F12D  LDX 02E2    si code trop grand ou négatif
F0A8  BNE F0E7    F130  BNE F16D    erreur
F0AA  LDX 02E1    F132  LDX 02E1    prendre code demandé
F0AD  CPX #20     F135  CPX #20     si code de controle,erreur
F0AF  BCC F0E7    F137  BCC F16D
F0B1  CPX #80     F139  CPX #80     si >128
F0B3  BCS F0E7    F13B  BCS F16D    erreur aussi
F0B5  LDA #02     F13D  LDA #02
```

```
F0B7  LDX #E3     F13F  LDX #E3     indexer #2E3
F0B9  JSR $F264   F141  JSR $F2F8   jeu de caractère 0 ou 1 ?
F0BC  BCS F0E7    F144  BCS F16D    non,erreur
F0BE  LDA #04     F146  LDA #04
F0C0  LDX #E5     F148  LDX #E5     indexer #2E5
F0C2  JSR $F264   F14A  JSR $F2F8   FB code plus petit que 4 ?
F0C5  BCS F0E7    F14D  BCS F16D    non, erreur
F0C7  LDA 0219    F14F  LDA 0219    prendre coordonnée horizontale
F0CA  CMP #E9     F152  CMP #EB     comparer à 235 (240-6 (largeur caractère) )
F0CC  BCS F0E7    F154  BCS F16D    si au delà,erreur
F0CE  LDA 021A    F156  LDA 021A    coordonnée verticale
F0D1  CMP #C1     F159  CMP #C1     comparer à 193 (200-8 (hauteur caractère) )
F0D3  BCS F0E7    F15B  BCS F16D    sauter si érreut
F0D5  JSR $F0EB   F15D  JSR $F171   calculer adresse du caractère
F0D8  JSR $F115   F160  JSR $F19B   afficher le caractère
F0DB  LDX 0219    F163  LDX 0219
F0DE  LDY 021A    F166  LDY 021A    prendre coordonnées curseur
F0E1  JSR $EFA6   F169  JSR $F049   récupérer adresse et motif du curseur
F0E4  JMP $F0EA   F16C  RTS
F0E7  INC 02E0    F16D  INC 02E0
F0EA  RTS         F170  RTS
```

CALCULER ADRESSE D'UN CARACTERE

```
F0EB  CLD         F171  CLD
F0EC  LDA 02E5    F172  LDA 02E5    prendre FB code
F0EF  STA 0212    F175  STA 0212    et sauver
F0F2  JSR $EDE3   F178  JSR $EEDE   et ajuster FB code
F0F5  LDA 02E1    F17B  LDA 02E1    prendre code ASCII
F0F8  STA 0C      F17E  STA 0C      et sauver
F0FA  LDA #00     F180  LDA #00     poids fort =0
F0FC  STA 0D      F182  STA 0D
F0FE  LDX #03     F184  LDX #03     3 décalages= *8
F100  ASL 0C      F186  ASL 0C
F102  ROL 0D      F188  ROL 0D
F104  DEX         F18A  DEX         calculer déplacement dans la table
F105  BNE F100    F18B  BNE F186    (code*8)
F107  LDA 02E3    F18D  LDA 02E3    prendre numéro du jeu
F10A  ASL A       F190  ASL A       *2=0 ou 2
F10B  ASL A       F191  ASL A       *4=0 ou 4 (#0000 ou #0400 car poids fort)
F10C  CLC         F192  CLC
F10D  ADC #98     F193  ADC #98     ajouter à adresse de base de la table
F10F  CLC         F195  CLC
```

```
F11Ø  ADC ØD      F196  ADC ØD      et à déplacement (le poids faible
F112  STA ØD      F198  STA ØD      ne bouge pas)
F114  RTS         F19A  RTS
```

AFFICHER UN CARACTERE

Entrée: #ØC-#ØD contient son adresse dans la table des caractères

Sortie: rien de spécial.

```
.............. F19B  CLD
F115  LDY #ØØ     F19C  LDY #ØØ
F117  STY ØF      F19E  STY ØF      indiquer premier octet de la définition
F119  LDA (ØC),Y  F1AØ  LDA (ØC),Y  saisir motif du caractère
F11B  STA ØE      F1A2  STA ØE      et sauver
F11D  JSR $F2C3   F1A4  JSR $F35D   sauver motif curseur
F12Ø  ROL ØE      F1A7  ROL ØE
F122  ROL ØE      F1A9  ROL ØE      décaler (éviter b7 et b6 inutiles)
F124  LDX #Ø6     F1AB  LDX #Ø6     il y a six points par octet
F126  ROL ØE      F1AD  ROL ØE      sortir un point
F128  BCC F12D    F1AF  BCC F1B4    si Ø,on ne fait rien
F12A  JSR $EF5B   F1B1  JSR $FØ24   afficher le point sans le pattern
F12D  JSR $FØØ4   F1B4  JSR $FØA1   et déplacer à droite
F13Ø  DEX         F1B7  DEX         et pour les six points
F131  BNE F126    F1B8  BNE F1AD
F133  JSR $F2D4   F1BA  JSR $F36E   récupérer motif 1er point
F136  JSR $EFE6   F1BD  JSR $FØ89   déplacer vers le bas
F139  LDY ØF      F1CØ  LDY ØF      prendre numéro de l'octet
F13B  INY         F1C2  INY         on prépare le suivant
F13C  CPY #Ø8     F1C3  CPY #Ø8     c'était le dernier ?
F13E  BNE F117    F1C5  BNE F19E    non,on recommence
F14Ø  RTS         F1C7  RTS
```

EVALUER 'COULEUR' D'UN POINT

Entrée: #2E1-#2E2: coordonnée horizontale du point demandé
#2E3-#2E4: coordonné verticale du point demandé

Sortie: #2E1-#2E2=Ø si le point est éteint ou #FFFF (soit -1) s'il est allumé.

Remarque: assez lent puisque nécessitant une division.

```
F141  LDA #F0      F1C8  LDA #F0      240
F143  LDX #E1      F1CA  LDX #E1      indexer #2E1
F145  JSR $F264    F1CC  JSR $F2F8    vérifier horizontal plus petit que 240
F148  BCS F17B     F1CF  BCS F200     non, erreur
F14A  LDA #C8      F1D1  LDA #C8      200
F14C  LDX #E3      F1D3  LDX #E3      indexer #2E3
F14E  JSR $F264    F1D5  JSR $F2F8    vérifier vertical plus petit que 200
F151  BCS F17B     F1D8  BCS F200     non, erreur
F153  LDX Ø2E1     F1DA  LDX Ø2E1
F156  STX Ø219     F1DD  STX Ø219
F159  LDY Ø2E3     F1E0  LDY Ø2E3
F15C  STY Ø21A     F1E3  STY Ø21A
F15F  JSR $EFA6    F1E6  JSR $F049    évaluer motif du point demandé
F162  LDY #ØØ      F1E9  LDY #ØØ
F164  LDA (1Ø),Y   F1EB  LDA (1Ø),Y   prendre sextet
F166  AND Ø215     F1ED  AND Ø215     et isoler le bit concerné
F169  BEQ F170     F1F0  BEQ F1F7     si éteint, retourner ØØ
F16B  LDA #FF      F1F2  LDA #FF      si allumé, retourner -1
F16D  JMP $F172    F1F4  JMP $F1F9
F170  LDA #ØØ      F1F7  LDA #ØØ      inutile (on vient de BEQ !)
F172  STA Ø2E1     F1F9  STA Ø2E1
F175  STA Ø2E2     F1FC  STA Ø2E2     sauver résultat
F178  JMP $F17E    F1FF  RTS
F17B  INC Ø2E0     F200  INC Ø2E0
F17E  RTS          F203  RTS
```

'PAPER' (COMMANDE)

Entrée: #2E1-#2E2 contient la valeur de la couleur de fond.

Sortie: rien de spécial

```
F17F  LDA #1Ø      F204  LDA #1Ø
F181  STA ØC       F206  STA ØC       masque pour attribut de fond= %ØØØ1ØBVR
F183  LDA #ØØ      F208  LDA #ØØ
F185  STA ØD       F20A  STA ØD       et à placer sur colonne Ø
F187  JSR $F197    F20C  JSR $F21C    et agir en conséquence
F18A  RTS          F20F  RTS
```

'INK' (COMMANDE)

Entrée: #2E1-#2E2 contient la couleur d'encre

Sortie: rien de spécial.

```
F18B  LDA #00     F210  LDA #00     masque pour attribut encre =%00000BVR
F18D  STA 0C      F212  STA 0C
F18F  LDA #01     F214  LDA #01     et à placer sur colonne 1
F191  STA 0D      F216  STA 0D
F193  JSR $F197   F218  JSR $F21C
F196  RTS         F21B  RTS
```

INK/PAPER SOUS ROUTINE

```
F197  LDA #08     F21C  LDA #08
F199  LDX #E1     F21E  LDX #E1     indexer #2E1
F19B  JSR $F264   F220  JSR $F2F8   vérifier couleur de 0 à 7
F19E  BCS F1E1    F223  BCS F264    non, erreur
F1A0  JSR $F2C3   F225  JSR $F35D   sauver #10-#11
F1A3  LDA 02E1    F228  LDA 02E1    prendre couleur
F1A6  ORA 0C      F22B  ORA 0C      et modifier selon attribut désiré
F1A8  STA 0202    F22D  STA 0202    et sauver
F1AB  LDX 021F    F230  LDX 021F    prendre drapeau de mode
F1AE  BNE F1C2    F233  BNE F247    sauter si HIRES
F1B0  LDX 0D      F235  LDX 0D      mode TEXT:prendre index
F1B2  STA 026B,X  F237  STA 026B,X  et sauver nouvel attribut dans registres
F1B5  LDA #A8     F23A  LDA #A8     calculer adresse ,soit #BBA8 ou #BBA9
F1B7  CLC         F23C  CLC
F1B8  ADC 0D      F23D  ADC 0D      ajouter colonne désirée
F1BA  TAX         F23F  TAX         poids faible dans X
F1BB  LDY #BB     F240  LDY #BB     poids fort dans X
F1BD  LDA #1B     F242  LDA #1B     et 27 lignes à colorier
F1BF  JMP $F1CC   F244  JMP $F251   (ou BNE) finir
F1C2  LDA #00     F247  LDA #00
F1C4  CLC         F249  CLC
F1C5  ADC 0D      F24A  ADC 0D
F1C7  TAX         F24C  TAX         calculer adresse de base (#A000 ou #A001)
F1C8  LDY #A0     F24D  LDY #A0     dans XY
F1CA  LDA #C8     F24F  LDA #C8     et 200 lignes à colorier
```

Colorier A lignes à l'adresse XY

```
F1CC  STA 0200    F251  STA 0200    sauver compteur ligne
F1CF  STX 10      F254  STX 10
F1D1  STY 11      F256  STY 11      et adresse de base
F1D3  LDA #01     F258  LDA #01
```

```
F1D5  STA Ø2Ø1    F25A  STA Ø2Ø1    indiquer un seul sextet
F1D8  JSR $F23A   F25D  JSR $F2CD   et faire comme FILL
F1DB  JSR $F2D4   F26Ø  JSR $F36E   récuperer #1Ø-#11
F1DE  JMP $F1E4   F263  RTS
F1E1  INC Ø2EØ    F264  INC Ø2EØ
F1E4  RTS         F267  RTS
```

'FILL' (COMMANDE)

Entrée: #2E1-#2E2 contient le nombre de lignes à remplir
#2E3-#2E4 contient le nombre de sextets à remplir
#2E5-#2E6 contient la valeur de remplissage.

Sortie: #2EØ incrémenté si erreur dans les paramètres.

```
F1E5  CLD         F268  CLD
.....             F269  LDA Ø2E3    prendre largeur en sextet
.....             F26C  STA Ø2Ø1    et sauver
.....             F26F  BEQ F2C9    si nul, erreur
.....             F271  LDY #ØØ     calculer position en sextet du curseur
.....             F273  LDA Ø219    prendre coordonnée curseur
.....             F276  SEC
.....             F277  SBC #Ø6     et enlever un sextet
.....             F279  BCC F27F    sortir si <Ø
.....             F27B  INY         sinon, sextet suivant
.....             F27C  JMP $F276   et on continue
.....             F27F  TYA         position en sextet dans A
.....             F28Ø  CLC
.....             F281  ADC Ø2E3    ajouter le nombre désiré
.....             F284  TAY         et on sauve
.....             F285  LDA Ø2E4
.....             F288  ADC #ØØ     ajuster poids fort aussi
.....             F28A  BNE F2C9    si négatif ou trop grand, erreur
.....             F28C  CPY #29     vérifier pas plus de 41 sextet par ligne...
.....             F28E  BCS F2C9    et erreur si oui
.....             F29Ø  LDA Ø2E6    prendre code de remplissage, poids fort
.....             F293  BNE F2C9    et erreur si négatif ou trop grand
F1E6  LDA Ø2E1    F295  LDA Ø2E1    prendre le nombre de lignes
F1E9  STA Ø2ØØ    F298  STA Ø2ØØ    et sauver
F1EC  BEQ F236    F29B  BEQ F2C9    si aucune ligne, erreur
F1EE  CLC         F29D  CLC
F1EF  ADC Ø21A    F29E  ADC Ø21A    plus coordonnée verticale
F1F2  TAY         F2A1  TAY         dans Y (limite basse)
```

```
F1F3  LDA Ø2E2    F2A2  LDA Ø2E2
F1F6  ADC #ØØ     F2A5  ADC #ØØ     poids fort aussi
F1F8  BNE F236    F2A7  BNE F2C9    si négatif ou déjà trop grand, erreur
F1FA  CPY #C9     F2A9  CPY #C9     verifier =< 2Ø1
F1FC  BCS F236    F2AB  BCS F2C9    non, erreur
F1FE  LDA Ø2E3    ..... ..........  prendre nombre de sextets
F2Ø1  STA Ø2Ø1    ..... ..........  et sauver
F2Ø4  BEQ F236    ..... ..........  si nul, erreur
F2Ø6  LDY #ØØ     ..... ..........  calculer position en septet du curseur
F2Ø8  LDA Ø219    ..... ..........  prendre coordonnée horizontale:
F2ØB  SEC         ..... ..........
F2ØC  SBC #Ø6     ..... ..........  on enlève un sextet
F2ØE  BCC F214    ..... ..........  si <Ø,on a trouvé, c'est dans Y
F21Ø  INY         ..... ..........  non,indiquer sextet suivant
F211  JMP $F2ØB   ..... ..........  et continuer
F214  TYA         ..... ..........  position dans A
F215  CLC         ..... ..........
F216  ADC Ø2E3    ..... ..........  ajouter position demandée
F219  TAY         ..... ..........  dans Y
F21A  LDA Ø2E4    ..... ..........
F21D  ADC #ØØ     ..... ..........  poids fort aussi
F21F  BNE F236    ..... ..........  si négatif ou déjà trop grand, erreur
F221  CPY #29     ..... ..........  et tester inférieur à 41
F223  BCS F236    ..... ..........  non, erreur
F225  LDA Ø2E6    ..... ..........  prendre poids fort du code de remplissage
F228  BNE F236    ..... ..........  si négatif ou trop grand, erreur
..... ..........  F2AD  CPY #C8     nouvelle coordonnée verticale en bas ?
..... ..........  F2AF  BNE F2B3    non, c'est bon
..... ..........  F2B1  LDY #ØØ     oui, alors en haut
..... ..........  F2B3  STY Ø21A    et sauver nouvelle coordonnée
F22A  LDA Ø2E5    F2B6  LDA Ø2E5    prendre code de remplissage
F22D  STA Ø2Ø2    F2B9  STA Ø2Ø2    et sauver pour travail
F23Ø  JSR $F23A   F2BC  JSR $F2CD   effectuer le FILL
..... ..........  F2BF  LDY Ø21A
..... ..........  F2C2  LDX Ø219
..... ..........  F2C5  JSR $FØ49    placer adresse et motif du nouveau curseur
F233  JMP $F239   F2C8  RTS
F236  INC Ø2EØ    F2C9  INC Ø2EØ
F239  RTS         F2CC  RTS
```

FILL (CALCUL ET AFFICHAGE)

Entrée: #1Ø-#11 contient l'adresse du premier octet à 'colorier'

```
      #200 contient le nombre de lignes à remplir
      #201 contient le nombre de sextets par ligne
      #202 contient la valeur de remplissage

.............. F2CD  CLD
F23A  LDA #202   F2CE  LDA #202   prendre code de remplissage
F23D  LDY #00    F2D1  LDY #00    indexer début
F23F  STA (10),Y F2D3  STA (10),Y sauver à l'écran
F241  INY        F2D5  INY        sextet suivant
F242  CPY #201   F2D6  CPY #201   vérifier pas ini
F245  BNE F23F   F2D9  BNE F2D3   non, on continue
F247  JSR $EFE6  F2DB  JSR $F989  oui, se placer ligne suivante
F24A  DEC #200   F2DE  DEC #200   vérifier pas fini
F24D  BNE F23A   F2E1  BNE F2CE   non, on repart
F24F  RTS        F2E3  RTS
```

TESTER PARAMETRE NON NUL ET ASSEZ PETIT

```
Entrée: A contient la valeur limite (exclue)
      X indexe le paramètre en page 2

Sortie: C=1 si erreur (paramètre trop grand ou nul)

F250  STA #204   F2E4  STA #204   sauver maxi permis
F253  LDA #201,X F2E7  LDA #201,X si poids fort non nul
F256  BNE F262   F2EA  BNE F2F6   alors trop grand ou négatif: erreur
F258  LDA #200,X F2EC  LDA #200,X prendre valeur
F25B  BEQ F262   F2EF  BEQ F2F6   si nulle, erreur aussi
F25D  CMP #204   F2F1  CMP #204   sinon,comparer à limite
F260  BCC F263   F2F4  BCC F2F7   ou...RTS !
F262  SEC        F2F6  SEC
F263  RTS        F2F7  RTS
```

TESTER PARAMETRE ASSEZ PETIT

```
Entrée: A contient la valeur (exclue) limite du parametre
      X indexe un paramétre en page 2.

Sortie: C=1 si paramètre trop grand.

F264  STA #204   F2F8  STA #204   sauver valeur limite
F267  LDA #201,X F2FB  LDA #201,X prendre poids fort
```

```
F26A  BNE F274    F2FE  BNE F308    si négatif ou déjà trop grand, erreur
F26C  LDA Ø200,X  F300  LDA Ø200,X
F26F  CMP Ø204    F303  CMP Ø204    sinon, comparer à valeur limite
F272  BCC F275    F306  BCC F309    ou RTS...
F274  SEC         F308  SEC
F275  RTS         F309  RTS
```

DRAW:VERIFICATION DES PARAMETRES

Entrée: paramètres en #Ø2E1-#Ø2E6

Sortie: C=1 si paramètres invalides, C=Ø sinon.

```
F276  LDA #Ø4     F30A  LDA #Ø4     FB code doit être inférieur à 4
F278  LDX #E5     F30C  LDX #E5     indexer FB code
F27A  JSR $F264   F30E  JSR $F2F8   et tester
F27D  BCS F2C2    F311  BCS F35C    sortir, C=1 si pas bon
F27F  LDA #ØØ     ..............
F281  STA Ø201    ..............
F284  STA Ø203    ..............    annuler poids forts
F287  LDA Ø219    ..............    prendre poistion horizontale curseur hires
F28A  STA Ø200    ..............    et sauver
F28D  LDA Ø21A    ..............    idem position verticale
F290  STA Ø202    ..............
F293  LDX #ØØ     ..............    indexer #200
F295  LDY #E1     ..............    indexer #2E1
F297  JSR $EC95   ..............    et faire addition
..............    F313  CLC         calculer deuxième extrémité de la droite
..............    F314  LDA Ø2E1    déplacement horizontal
..............    F317  ADC Ø219    plus position curseur
..............    F31A  STA Ø200    et on sauve
..............    F31D  LDA Ø2E2
..............    F320  ADC #ØØ     poids fort aussi
..............    F322  STA Ø201    et on sauve
..............    F325  LDX #ØØ     indexer #200-#201
F29A  LDA #F0     F327  LDA #F0     pas plus de 240
F29C  JSR $F264   F329  JSR $F2F8   et tester
F29F  BCS F2C2    F32C  BCS F35C    si pas bon, on sort
F2A1  LDX #Ø2     ..............    indexer #202-#203
F2A3  LDY #E3     ..............    à ajouter à #2E3-#2E4
F2A5  JSR $EC95   ..............    et faire l'addition
..............    F32E  CLC         calculer coordonnée verticale
..............    F32F  LDA Ø2E3    de l'extrémite de la droite
```

```
.............. F332  ADC #21A
.............. F335  STA #202    dans #202-#203
.............. F338  LDA #2E4
.............. F33B  ADC #00
.............. F33D  STA #203
.............. F340  LDX #02     indexer #202-#203
F2A8 LDA #C8    F342  LDA #C8     pas plus de 200
F2AA JSR $F264  F344  JSR $F2F8   et faire le test
F2AD BCS F2C2   F347  BCS F35C    sortir si trop loin
F2AF LDA #2E5   F349  LDA #2E5    prendre FB code
F2B2 STA #212   F34C  STA #212    et le sauver
F2B5 LDA #200   F34F  LDA #200    récupérer nouvelle position du curseur
F2B8 STA #219   F352  STA #219    horizontale
F2BB LDA #202   F355  LDA #202
F2BE STA #21A   F358  STA #21A    puis horizontale
F2C1 CLC        F35B  CLC         inutile:indiquer paramètres valides.
F2C2 RTS        F35C  RTS
```

SAUVER ADRESSE ET MASQUE COURANT

```
F2C3 LDA 10     F35D  LDA 10      prendre pointeur adresse du curseur
F2C5 STA #216   F35F  STA #216    et le sauver (poids faible)
F2C8 LDA 11     F362  LDA 11
F2CA STA #217   F364  STA #217    idem poids fort
F2CD LDA #215   F367  LDA #215    sauver aussi motif courant
F2D0 STA #218   F36A  STA #218
F2D3 RTS        F36D  RTS
```

RECUPERER ADRESSE ET MASQUE COURANT

```
F2D4 LDA #216   F36E  LDA #216
F2D7 STA 10     F371  STA 10
F2D9 LDA #217   F373  LDA #217
F2DC STA 11     F376  STA 11      récupérer pointeur adresse curseur
F2DE LDA #218   F378  LDA #218
F2E1 STA #215   F37B  STA #215    et le motif courant
F2E4 RTS        F37E  RTS
```

'CIRCLE' (COMMANDE)

Entrée: #2E1-#2E2 contient le rayon

#2E3-#2E4 contient le FB code

Sortie: #2E0 a été incrémenté si le cercle sortait de l'écran (il n'est dans ce cas pas tracé), et sinon le cercle a été tracé.
#10-#11 et #215 ne sont pas affectés.

```
.............. F37F CLD         inutile
F2E5 LDA 02E2  F380 LDA 02E2    poids fort du rayon
F2E8 BNE F32D  F383 BNE F3C2    si rayon négatif ou > 255, erreur
F2EA LDA 02E1  F385 LDA 02E1    poids faible du rayon
F2ED BEQ F32D  F388 BEQ F3C2    si rayon nul, erreur
F2EF LDA 0219  F38A LDA 0219    si le curseur n'est pas assez loin
F2F2 CMP 02E1  F38D CMP 02E1    le cercle sort à gauche
F2F5 BCC F32D  F390 BCC F3C2    et erreur
F2F7 CLC       F392 CLC
F2F8 ADC 02E1  F393 ADC 02E1    calculer point extrême droite
F2FB CMP #F0   F396 CMP #F0     si le cercle sort à droite
F2FD BCS F32D  F398 BCS F3C2    erreur evidemment
F2FF LDA 021A  F39A LDA 021A
F302 CMP 02E1  F39D CMP 02E1    si le curseur est trop haut,
F305 BCC F32D  F3A0 BCC F3C2    le cercle sort en haut et erreur
F307 CLC       F3A2 CLC
F308 ADC 02E1  F3A3 ADC 02E1    calculer point le plus bas
F30B CMP #C8   F3A6 CMP #C8
F30D BCS F32D  F3A8 BCS F3C2    et erreur si le cercle sort en bas
F30F LDX #E3   F3AA LDX #E3     indexer #2E3 (FB code)
F311 LDA #04   F3AC LDA #04     que l'on veut plus petit que 4
F313 JSR $F264 F3AE JSR $F2F8   effectuer le test
F316 BCS F32D  F3B1 BCS F3C2    et erreur si pas bon
F318 LDA 02E3  F3B3 LDA 02E3    prendre FB code
F31B STA 0212  F3B6 STA 0212    et le sauver
F31E JSR $EDE3 F3B9 JSR $EED8   ajuster FB code
F321 LDA 0213  ..............
F324 STA 0214  .............. sauver le registre motif
F327 JSR $F331 F3BC JSR $F3C6   calcul et affichage du cercle
F32A JMP $F330 F3BF JMP $F3C5   ou...RTS: c'est fini
F32D INC 02E0  F3C2 INC 02E0    indiquer erreur
F330 RTS       F3C5 RTS
```

CIRCLE (CALCUL ET AFFICHAGE)

Entrée: les paramètres sont positionnés comme pour la routine précédente, mais leur validité n'est pas testée.

Sortie: le cercle est tracé... les paramètres du curseur n'ont pas changés (#10-#11 et #215)

Algorithme: à partir des coordonnées du centre (Cx et Cy) on calcule les coordonnées du premier point, le sommet du cercle (Sx et Sy), pour placer le curseur en absolu. A partir de maintenant, on va travailler en relatif, et d'abord calculer les coordonnées du sommet en relatif (S'x=0 et S'y=R).

On va ensuite calculer Dx et Dy (les déplacements entre deux points successifs), et ainsi obtenir les nouvelles coordonnées relatives (S'x=S'x+Dx, S'y=S'y+Dy). Pour s'arrêter, on teste si on n'est pas revenu au point de départ (coordonnées 0,R).

Les calculs sont faits avec un octet contenant la partie entière de la coordonnée, et un octet contenant la partie décimale, ou fractionnaire. Un nouveau point n'est affiché que si la partie entière d'une des deux coordonnées a changée.

Le plus dur est de calculer correctement (quoiqu'une grande précision ne soit pas nécessaire) et rapidement surtout Dx et Dy, sans passer par de très long calculs trigonométriques en virgule flottante... Les détails de ce calcul sont expliqués à la routine "calcul de Dx ou DY".

Variables systèmes:

```
#0C  :travail Dx ou Dy,fracionnaire
#0D  :idem partie entière
#0E  :exposant du rayon
#0F  :0:point déjà affiché, 1:il faut afficher
#021B:S'x fractionnaire
#021C:S'x entière
#021D:S'y fractionnaire
#021E:S'y entière
```

```
F331  JSR $F2C3    F3C6  JSR $F35D   sauver #10-#11-#215
F334  LDA 021A     F3C9  LDA 021A    position verticale
F337  SEC          F3CC  SEC
F338  SBC 02E1     F3CD  SBC 02E1    -rayon=verticale du plus haut point (Sy)
F33B  TAY          F3D0  TAY         dans Y
F33C  LDX 0219     F3D1  LDX 0219    on prend la coordonnée du centre (Sx=Cx)
F33F  JSR $EFA6    F3D4  JSR $F049   et calcul adresse du sommet du cercle
F342  LDA 02E1     F3D7  LDA 02E1    prendre rayon
F345  STA 0F       F3DA  STA 0F      le sauver pour travail
F347  JSR $F3F0    F3DC  JSR $F485   et calculer en #0E son exposant base 2
F34A  LDA #80      F3DF  LDA #80     partie décimale= #80 (1/2)
F34C  STA 021B     F3E1  STA 021B    pour coordonnée horizontale
F34F  STA 021D     F3E4  STA 021D    et coordonnée verticale
```

```
F352  LDA #00     F3E7  LDA #00     abscisse relative (S'x)=0
F354  STA 021C    F3E9  STA 021C
F357  LDA 02E1    F3EC  LDA 02E1    et ordonnée relative (S'y)=rayon
F35A  STA 021E    F3EF  STA 021E
F35D  LDA #00     F3F2  LDA #00
F35F  STA 0F      F3F4  STA 0F      indiquer pas d'affichage
F361  JSR $F37F   F3F6  JSR $F414   calculer prochaine coordonnée S'x
F364  JSR $F3AF   F3F9  JSR $F444   et S'y
F367  LDA 0F      F3FC  LDA 0F      si une des deux a changé,on affiche
F369  BEQ F36E    F3FE  BEQ F403    sinon on n'affiche pas
F36B  JSR $EF4D   F400  JSR $F016   afficher le point
F36E  LDA 021C    F403  LDA 021C    si S'x non nul,ce n'est pas la fin
F371  BNE F35D    F406  BNE F3F2    et on recommence
F373  LDA 021E    F408  LDA 021E    sinon,est-ce le point le plus bas ?
F376  CMP 02E1    F40B  CMP 02E1
F379  BNE F35D    F40E  BNE F3F2    oui, on recommence
F37B  JSR $F2D4   F410  JSR $F36E   non, on récupère les registres
F37E  RTS         F413  RTS        et on sort
```

CALCULER NOUVELLE POSITION HORIZONTALE

```
F37F  LDA 021D    F414  LDA 021D    prendre S'y décimal
F382  LDX 021E    F417  LDX 021E    et partie entière
F385  JSR $F3DF   F41A  JSR $F474   calculer Dx dans #0C-#0D
F388  LDA 0C      F41D  LDA 0C      prendre partie décimale
F38A  CLC         F41F  CLC
F38B  ADC 021B    F420  ADC 021B    et ajouter à S'x
F38E  STA 021B    F423  STA 021B
F391  LDA 021C    F426  LDA 021C    idem pour partie entière
F394  STA 0C      F429  STA 0C
F396  ADC 0D      F42B  ADC 0D
F398  STA 021C    F42D  STA 021C
F39B  CMP 0C      F430  CMP 0C      la partie entière a-t-elle changée ?
F39D  BEQ F3AE    F432  BEQ F443    non, on sort
F39F  BCS F3A7    F434  BCS F43C    sauter si déplacement à gauche
F3A1  JSR $F004   F436  JSR $F0A1   déplacer vers la droite
F3A4  JMP $F3AA   F439  JMP $F43F   et indiquer affichage
F3A7  JSR $F015   F43C  JSR $F0B2   déplacer vers la gauche
F3AA  LDA #01     F43F  LDA #01
F3AC  STA 0F      F441  STA 0F      indiquer qu'il faut afficher
F3AE  RTS         F443  RTS
```

CALCULER NOUVELLE POSITION VERTICALE

```
F3AF  LDA Ø21B    F444  LDA Ø21B    prendre S'x partie décimale
F3B2  LDX Ø21C    F447  LDX Ø21C    et partie entière
F3B5  JSR $F3DF   F44A  JSR $F474   et caluler Dy
F3B8  SEC         F44D  SEC
F3B9  LDA Ø21D    F44E  LDA Ø21D    retrancher de S'y partie décimale
F3BC  SBC ØC      F451  SBC ØC
F3BE  STA Ø21D    F453  STA Ø21D
F3C1  LDA Ø21E    F456  LDA Ø21E    idem poids fort
F3C4  STA ØC      F459  STA ØC
F3C6  SBC ØD      F45B  SBC ØD
F3C8  STA Ø21E    F45D  STA Ø21E
F3CB  CMP ØC      F46Ø  CMP ØC      est-ce le même point ?
F3CD  BEQ F3DE    F462  BEQ F473    oui,on sort
F3CF  BCS F3D7    F464  BCS F46C    sauter si point au dessus
F3D1  JSR $EFE6   F466  JSR $FØ89   décaler vers le bas
F3D4  JMP $F3DA   F469  JMP $F46F   et finir
F3D7  JSR $EFF5   F46C  JSR $FØ95   décaler vers le haut
F3DA  LDA #Ø1     F46F  LDA #Ø1     indiquer qu'il faut afficher
F3DC  STA ØF      F471  STA ØF
F3DE  RTS         F473  RTS
```

CALCULER Dx ou Dy

Entrée: AX contient la valeur d'une coordonnée (relative)
E contient l'exposant du rayon.

Sortie: #ØC-#ØD contient la différentielle de la coordonnée
Y inchangé.

Principe: Dx=S'y/A (ou Dy=S'x/A).

Pour avoir une précision suffisante, de sorte que les points du cercle soient contigus, il faut que A soit du même ordre que le rayon. En fait, A est approximé par la puissance de deux immédiatement supérieure au rayon (voir routine suivante), de sorte que la division par 2^n se ramène tout naturellement à n décalages à droite.

De plus, S'x et S'y étant signés, il faut prendre garde, lors des divisions, à réincorporer le signe au poids fort lors des décalages.

Mathématique: La formule est en fait tirée d'un formule plus générale, valable pour une ellipse quelconque d'équation:
(X/A)^2+(Y/B)^2=1 (A,B > Ø)

donne:

X' = X + e*A*A*Y

Y' = Y - e*B*B*X'

Pour un cercle, l'exentricité (e) est bien entendu égale à 1, et A=B= rayon.

Remarque: cet algorithme de calcul de différentielle est très puissant, et est exploitable sous BASIC. Le programme suivant trace un cercle de rayon 50:

```
10 X=50:Y=0
20 X=X+Y/64
30 Y=Y-X/64
40 CURSET 120+X,100+Y,1
50 GOTO 20
```

Il suffit de faire varier le coefficient A dans le calcul de la différentielle pour tracer des ellipses... ou des vrais cercles !

```
F3DF  STA 0C      F474  STA 0C
F3E1  STX 0D      F476  STX 0D      sauver la coordonnée
F3E3  LDX 0E      F478  LDX 0E      compter les décalages (diviser par 2^(#0E)
F3E5  LDA 0D      F47A  LDA 0D      prendre signe
F3E7  ROL A       F47C  ROL A       dans C
F3E8  ROR 0D      F47D  ROR 0D      diviser par deux, y compris signe
F3EA  ROR 0C      F47F  ROR 0C      idem poids faible
F3EC  DEX         F481  DEX
F3ED  BNE F3E5    F482  BNE F47A    et continuer la division.
F3EF  RTS         F484  RTS
```

CALCULER L'ORDRE DU RAYON

Principe: on calcule dans A les puissances successives de 2, et on compare au rayon. Attention: cette routine donne des résultats erronés si le rayon est supérieur à 128, ce qui ne se produit jamais en temps normal il est vrai.

Attention:l'exposant calculé est celui immédiatement supérieur au nombre (ex: pour #F=6,#E=3)

```
F3F0  INC 0F      F485  INC 0F      incrémenter le rayon
F3F2  LDA #00     F487  LDA #00     initialiser exposant
F3F4  STA 0E      F489  STA 0E
F3F6  LDA #01     F48B  LDA #01     commencer exposant=0
F3F8  ASL A       F48D  ASL A       décaler valeur de référence
F3F9  INC 0E      F48E  INC 0E      exposant+1
F3FB  CMP 0F      F490  CMP 0F      et comparer
```

```
F3FD  BCC F3F8    F492  BCC F48D    si toujours supérieur, on recommence
F3FF  RTS         F494  RTS
```

TABLE DE VECTEURS

Remarque: hélas, cette table n'apporte aucune aide pour résoudre les problèmes de compatibilité V1.Ø/V1.1. Son utilité est plus que douteuse...

```
F4ØØ  JMP $F7ØØ  ..............  AY=4Ø‡A
F4Ø3  JMP $F7CB  ..............  Allumer ou eteindre le curseur
F4Ø6  JMP $F535  ..............  accès GI 8912
F4Ø9  JMP $F73F  ..............  afficher à l'écran X
F4ØC  JMP $F7EØ  ..............  générer les caractères alternés
F4ØF  JMP $F43C  ..............  gérer le clavier
F412  JMP $FA85  ..............  PING
F415  JMP $FA9B  ..............  SHOOT
F418  JMP $FAB1  ..............  EXPLODE
F41B  JMP $FAC7  ..............  ZAP
F41E  JMP $FB26  ..............  SOUND
F421  JMP $FBB6  ..............  PLAY
F424  JMP $FBFE  ..............  MUSIC
F427  JMP $F923  ..............  TEXT
F42A  JMP $F8E3  ..............  HIRES
F42D  JMP $F84A  ..............  RESET
F43Ø  JMP $F882  ..............  NMI
F433  JMP $F57B  ..............  envoyer A sur l'imprimante
F436  JMP $F82F  ..............  Affichage ligne Ø
F439  JMP $F96Ø  ..............  Configuration VIA
```

L-GESTION DU CLAVIER

1) Procédures standards de test de touche

L'organisation du clavier est classique, puisqu'organisé selon une matrice 8x8, dont le détail des connexions est donné dans le chapitre entrée/sortie.

Ainsi, les colonnes sont accessibles comme 8 bits distincts, que l'on sélectionne à travers le PSG 8912 par le port A du VIA.

Les lignes sont quant à elles multiplexées, de sorte qu'elles n'occuppent que 3 bits du port B du VIA.

Le principe de lecture d'une touche est simple: on sélectionne une colonne, en mettant le bit correspondant à Ø, les autres restant à 1 pour ne pas sélectionner plusieurs touches à la fois. Il suffit ensuite de lire le résultat (connectée ou non) sur la ligne voulue.

a) Activer une colonne

Exemple:activer la colonne 2.

```
LDA #14         ;Préparer accès au PORT A du PSG 8912
STA VIAORA      ;Préférable à VIADRA pour ne pas toucher à CA1 et CA2
LDA #%11101110 ;CA2=1,CB2=1: indiquer adresse
STA VIAPCR
LDA #%11001100 ;CA2=Ø,CB2=Ø: latcher adresse
LDA #%11111011 ;masque pour colonne 2
STA VIAORA
LDA #%11101100 ;CA2=1,CB2=Ø: indiquer donnée
STA VIAPCR
LDA #%11001100 ;CA2=Ø:CB2=Ø: latcher donnée
STA VIAPCR
```

Remarques: la méthode décrite ici est en fait une procédure standard d'accès au PSG 8912, dont le bus d'adresse/donnée est multiplexé.

Cette méthode est un peu simplifiée puisqu'elle force à Ø le bit de sens de transition (bØ et b4 de VIAPCR), ce qui est peu génant. Se reporter à la routine de la V1.1 (#F59Ø) pour avoir un procédé plus rigoureux.

Le port A du GI 8912 est latché, ce qui veut dire que la donnée restera présente tant qu'on ne la remplacera pas explicitement par une autre. Inutile donc de répéter cette procédure pour acceder à plusieurs lignes d'une même colonne.

b) tester une ligne

Exemple: tester la ligne 3 de la colonne couramment activée

```
LDA #%10010011 ;b0,b1,b2=No de ligne, b7, b6 et b4 sont placés de manière à
STA VIADRB     ;ne pas activer le strobe, la sortie K7 ou le relais.
LDA #%00001000 ;préparer masque pour 'réponse' et attendre 2 microsecondes
AND VIADRB     ;tester la réponse
BEQ NONENFONCE
BNE ENFONCE
```

Remarques: le port B du VIA étant presqu'entièrement configuré en sortie, et l'activation des lignes n'occupant que 4 bits, il faut prendre garde à ne pas écrire n'importe quoi dans les ligne configurées en sortie, et notamment, laisser le strobe de l'imprimante (b4) à 1, la sortie cassette (b7) à 1 aussi et le relais ouvert à 0 (b6). Pour être tout à fait rigoureux, il faudrait en fait la séquence:

```
LDA VIADRB
AND #%11010000
ORA #%00000LLL
STA VIADRB
etc...
```

D'autre part, le multiplexeur utilisé a une réponse , pour certains Orics et selon la température, assez lente. Il est donc nécessaire d'attendre un peu entre l'écriture de la ligne et la lecture du résultat. Les 2 microsecondes du LDA #$00 suffisent dans tous les cas. Certains concepteurs de jeux l'ont négligé (toute la gamme Softek par exemple), et leur gestion du clavier est inopérante.

2) Les variables systèmes utilisées

La gestion du clavier utilise la zone #208-#211. Voici la structure des variables systèmes,consultables directement en BASIC ou assembleur, à la seule condition que les IRQ's soient autorisées et non détournées:

#208: %V0CCCLLL sert de 'Latche' pour les touches normales, et contient l'adresse ligne/colonne. Contient #38 si aucune touche n'est pressée.

#209: %V0CCCLLL idem à #208 mais pour la colonne 4 (shift,funct,Ctrl...)

#20A: motif de la colonne, touches normales. Le bit de la colonne considéré est à 0, les autrea à 1.

#20B: idem à #20A mais pour la colonne 4.
#20C: %M1111111 le bit M indique le mode majuscule (à 1) ou minuscule (à Ø). Les autres bits doivent être laissés à 1.
#20D: %ØØCCCØØØ compteur de colonne (travail).
#20E: compteur de répétition.
#20F: utilisé par la V1.Ø seulement pour la routine d'E/S avec le PSG 8912.
#210: %VØØØØLLL travail, sauver la valeur courante de la ligne.
#211: %ØØØØØLLL travail, première ligne à tester dans la colonne.

Sur l'Atmos, les variables #24E et #24F contiennent de plus les valeurs de répétition.

Abréviations utilisées:

CCC indique que les Bits concernés contiennent un numéro de colonne.
LLL idem ci-dessus mais pour un numéro de ligne.
V est un indicateur qui est à un si une touche est pressée, à Ø sinon.

GESTION DU CLAVIER

Entrée: la zone #2Ø8-#2ØE doit être cohérente pour gérer correctement la répétition des touches.

Sortie: si b7 de X est à 1, alors le codes ASCII de la touche est dans bØ-b6 de X.
si b7 de X est à Ø, aucune touche n'est pressée.
A,Y et P (sauf N et Z) ne sont pas touchés.

Principe: si une touche était pressée précédemment, la routine commence par tester cette touche, en vue d'une éventuelle répétition. Sinon tout le clavier est balayé, colonne par colonne.

Différences entre la V1.1 et la V1.Ø: sur la V1.1, les valeurs de répétitions ont été vectorisés en #24E, #24F. Ceci permet de jouer facilement sur la vitesse de répétition. Exemple:DOKE #24E,#1Ø8.

```
F43C  PHA          F495  PHA        sauvegarde des registres
F43D  PHP          F496  PHA
F43E  TYA          F497  TYA        inutile
F43F  PHA          F498  PHA        pour le Basic.
F44Ø  CLD          F499  CLD        par précaution...
F441  LDA Ø2Ø8     F49A  LDA Ø2Ø8   brancher si lors de la précedente
F444  BPL F463     F49D  BPL F4BD   scrutation, pas de touche pressée.
F446  AND #87      F49F  AND #87    isoler le numéro de ligne
```

```
F448  STA Ø21Ø     F4A1  STA Ø21Ø     et le sauver en #21Ø
F44B  LDX Ø20A     F4A4  LDX Ø20A     prendre la colonne dans X
F44E  JSR $F5Ø6    F4A7  JSR $F561    scruter toute les lignes de la colonne
F451  CMP Ø21Ø     F4AA  CMP Ø21Ø     -Si c'est la m€me ligne,
F454  BNE F463     F4AD  BNE F4BD       c'est donc la m€me touche,
F456  DEC Ø20E     F4AF  DEC Ø20E       on décrémente alors la répétition
F459  BNE F48C     F4B2  BNE F4E7       et on sort avec X)Ø si il faut attendre.
F45B  LDA #Ø4      F4B4  LDA Ø24F       sinon on recharge le compteur
F45D  STA Ø20E     F4B7  STA Ø20E       avec la valeur de répétition rapide
F46Ø  JMP $F46B    F4BA  JMP $F4C6      puis on valide encore la touche.
F463  LDA #2Ø      F4BD  LDA Ø24E     -Si la touche n'est plus enfoncée,
F465  STA Ø20E     F4CØ  STA Ø20E       on charge le compteur pour attente
F468  JSR $F4C8    F4C3  JSR $F523      et on scrute tout le clavier.
F46B  JSR $F494    F4C6  JSR $F4EF    on trouve le code ASCII de la touche -) A
F46E  TAX          F4C9  TAX          on sauve le code dans X, et on positionne N
F46F  BPL F48E     F4CA  BPL F4E9     si aucune touche n'est enfoncée on sort X)Ø
F471  PHA          F4CC  PHA          sauver le code sur la pile
F472  LDA Ø26A     F4CD  LDA Ø26A
F475  AND #Ø8      F4DØ  AND #Ø8      Bip sonore ?
F477  BNE F488     F4D2  BNE F4E3     non, sortir sans oublier de récupérer A
F479  PLA          F4D4  PLA          oui, récupérer le code (inutile,il est ds X)

F47A  (HA          F4D5  PHA          sans toucher à la pile
F47B  CMP #AØ      F4D6  CMP #AØ      tester si caractère de controle:
F47D  BCC F485     F4D8  BCC F4EØ
F47F  JSR $FAFA    F4DA  JSR $FB14     non, émettre un son
F482  JMP $F488    F4DD  JMP $F4E3     et finir
F485  JSR $FB1Ø    F4EØ  JSR $FB2A     oui, émettre ... un autre son
F488  PLA          F4E3  PLA          en tous cas récuperer le code
F489  JMP $F48E    F4E4  JMP $F4E9    et finir
F48C  LDA #ØØ      F4E7  LDA #ØØ      indiquer pas de touche (b7=Ø)
F48E  TAX          F4E9  TAX          mettre code dans X (c'est lui qu'on teste)
F48F  PLA          F4EA  PLA
F49Ø  TAY          F4EB  TAY          récuperer les registres...
F491  PLP          F4EC  PLP
F492  PLA          F4ED  PLA
F493  RTS          F4EE  RTS          et finir.
```

TROUVER LE CODE ASCII

Entrée: #2Ø8 doit contenir le code de la touche, #2Ø9 le code pour shift ou Ctrl, et #2ØC le masque majuscule/minuscule.
Sortie: A contient le code ASCII (b7 à 1), ou b7 à Ø si la touche n'est pas pressée. Tous les registres sont affectés.

Principe: cette routine, trés courte, est en fait assez compliquée. Pour bien la comprendre, il est important d'avoir assimilé la structure de la variable système #208 ainsi que la structure de la table #FF70-#FF78, donnée ci-dessous.

Le clavier est organisé en 8 colonnes de 8 lignes, cela fait donc 64 touches possibles,le double avec les touches shiftées. En #208,les 6 bits de poids faibles représentent l'adresse de la touche (3 bits de ligne et 3 bits de colonne, ce qui fait bien 64 possibilités.

Ces bits représentent en fait l'adresse de la touche dans la table. Si la touche est shiftée, il faut ajouter #40, soit 64 pour accéder au code de la touche.

Le code de la table est le code ASCII pour les majuscules et tous les caractères qui ne dépendent pas du passage minuscules/majuscules.

Les minuscules sont stockées code ASCII (majuscule)+#A0, ceci pour permettre assez facilement de retrouver des majuscules lorsqu'on est en mode minuscule, avec l'aide du shift, ou de toujours trouver des majuscules en mode majuscule,même shifté: par exemple, 'o' est stocké #EF, soit #4F+#A0. En éliminant b7, on a #6F, soit le code de o minuscule. Si on est en mode majuscule, on enlève #20 et on obtient #4F, le code de 'O' majuscule.

Le Bit 7 semble inutile, il ne l'est pas: c'est lui qui dit (Cf AND 20C) si on doit ou non forcer la majuscule.

```
F494  LDA 0209    F4EF  LDA 0209    prendre le code des touches de controle
F497  TAY         F4F2  TAY         puis dans Y (pourquoi pas  LDY 0209 !!?)
F498  LDA #00     F4F3  LDA #00     déplacement dans la table=0
F49A  CPY #A4     F4F5  CPY #A4     tester shift gauche
F49C  BEQ F4A2    F4F7  BEQ F4FD    brancher si shift gauche
F49E  CPY #A7     F4F9  CPY #A7     et shift droit
F4A0  BNE F4A5    F4FB  BNE F500
F4A2  CLC         F4FD  CLC         si shift, déplacement dans la table=#40
F4A3  ADC #40     F4FE  ADC #40     (pourquoi pas LDA #40 !!?)
F4A5  CLC         F500  CLC         ajouter à l'adresse de la touche
F4A6  ADC 0208    F501  ADC 0208    si vide, #38+#40=#78
F4A9  BPL F4C7    F504  BPL F522    et sortir A>0, N=0 si pas de touche
F4AB  AND #7F     F506  AND #7F     éliminer Bit 7
F4AD  TAX         F508  TAX         et préparer indexation
F4AE  LDA FF70,X  F509  LDA FF78,X  prendre code de la touche
F4B1  AND 020C    F50C  AND 020C    ET avec Min/Maj.Bit 7=0 si mode minuscule
F4B4  BPL F4B9    F50F  BPL F514    ou majuscule trouvée dans la table
F4B6  SEC         F511  SEC         si minuscule et mode majuscule,
F4B7  SBC #20     F512  SBC #20     retrouver majuscule
F4B9  AND #7F     F514  AND #7F     éliminer Bit7 (minuscule et mode majuscule)
F4BB  CPY #A2     F516  CPY #A2     tester si Ctrl
F4BD  BNE F4C5    F518  BNE F520    non, sortir.
F4BF  CMP #40     F51A  CMP #40     oui, est-ce une lettre ?
```

```
F4C1  BMI F4C5    F51C  BMI F520     non, sortir.
F4C3  AND #1F     F51E  AND #1F      oui, ramener modulo #20
F4C5  ORA #80     F520  ORA #80     Bit 7 à 1: touche valide.(N=1)
F4C7  RTS         F522  RTS
```

BALAYER TOUT LE CLAVIER

Entrée: rien ...

Sortie: #208, #209, #20A, #20B correctement positionnés. b7 de #208 à un si la touche est pressée. Tous les registres sont affectés. A noter que, si plusieurs touches sont pressées simultanément, la routine prendra en compte la dernière détectée, c'est à dire la plus près de la colonne 0, ligne 0. Ce phénomène est difficile à mettre en évidence à cause de la gestion de la répétition, il est difficile d'appuyer réellement simultanément sur plusieurs touches à la fois.

```
F4C8  LDA #38     F523  LDA #38     initialiser les variables et pointeurs:
F4CA  STA 020D    F525  STA 020D     Pointeur de travail sur colonne 7.
F4CD  STA 0208    F528  STA 0208     Bit 7=0, indique pas de touche préssée.
F4D0  STA 0209    F52B  STA 0209     Bit 7=0, indique pas shift,ctrl...
F4D3  LDA #7F     F52E  LDA #7F      A=%01111111,motif de test pour colonne 7.
F4D5  PHA         F530  PHA         Sauver le motif colonne sur la pile.
F4D6  PLA         F531  PLA         Récuperer motif colonne,
F4D7  PHA         F532  PHA          tout en le laissant sur la pile.
F4D8  TAX         F533  TAX          le mettre dans X (passage de paramètre)
F4D9  LDA #07     F534  LDA #07      Indiquer commencer par colonne 7.
F4DB  JSR $F506   F536  JSR $F561    Scruter la colonne à partir de la ligne 7.
F4DE  ORA 020D    F539  ORA 020D     rajouter le numéro de la colonne
F4E1  BPL F4F5    F53C  BPL F550     si pas de touche, colonne suivante.
F4E3  LDX #00     F53E  LDX #00     préparer index pour touche normalle.
F4E5  LDY #20     F540  LDY #20     Y=colonne No 4.
F4E7  CPY 020D    F542  CPY 020D    si ce n'est pas elle, sauter (pas shift...)
F4EA  BNE F4ED    F545  BNE F548
F4EC  INX         F547  INX         indexer pour #209 et #20B.
F4ED  STA 0208,X  F548  STA 0208,X  sauver code de la touche.
F4F0  PLA         F54B  PLA
F4F1  PHA         F54C  PHA         récupérer motif colonne,
F4F2  STA 020A,X  F54D  STA 020A,X  et le sauver aussi.
F4F5  SEC         F550  SEC         préparer C pour rotation
F4F6  PLA         F551  PLA         prendre motif de la colonne,
F4F7  ROR A       F552  ROR A       passer colonne suivante (motif)
F4F8  PHA         F553  PHA         et resauver le motif.
F4F9  SEC         F554  SEC         passer colonne suivante (numéro)
```

```
F4FA  LDA Ø20D    F555  LDA Ø20D
F4FD  SBC #Ø8     F558  SBC #Ø8
F4FF  STA Ø20D    F55A  STA Ø20D    et actualiser le pointeur.
F5Ø2  BPL F4D6    F55D  BPL F531    si la fin ,A vaut #F8,donc N=1.
F5Ø4  PLA         F55F  PLA         Ajuster la pile. A vaut #FF,
F5Ø5  RTS         F56Ø  RTS          Çà peut servir !!.
```

BALAYER UNE COLONNE

Entrée: motif de la colonne dans X, première ligne scrutée dans A.

Sortie: A=Ø si aucune touche de la colonne n'est pressée, ou avec dans A le numéro de la ligne de la touche pressée, le Bit 7 est alors à un.

Bogue: le ORA #B8 explique pourquoi il est impossible en mode direct de fermer le relais (voir procédures standards). On peut toutefois y parvenir en mettant la ligne en entrée (POKE #3Ø2,#B7)

```
F5Ø6  PHA         F561  PHA         Sauver la ligne
F5Ø7  LDA #ØE     F562  LDA #ØE     indiquer registre d'E/S du 8912,
F5Ø9  JSR $F535   F564  JSR $F59Ø   et envoyer le motif de la colonne.
F5ØC  PLA         F567  PLA         récupérer la ligne à tester,
F5ØD  AND #Ø7     F568  AND #Ø7     isoler son numéro (surtout annuler Bit 7)
F5ØF  TAX         F56A  TAX         et le sauver dans X qui sert de pointeur.
F51Ø  STA Ø211    F56B  STA Ø211    et dans #211 pour savoir où s'arrêter.
F513  ORA #B8     F56E  ORA #B8     Forcer le strobe à 1, le relais ouvert.
F515  STA Ø3ØØ    F57Ø  STA Ø3ØØ    et envoyer la colonne au multiplexeur
F518  LDY #Ø4     F573  LDY #Ø4     attendre sa réponse
F51A  DEY         F575  DEY         (boucle de 22 µs, assez importante.
F51B  BNE F51A    F576  BNE F575    1 ou 2 µs suffisent, prudence de bon aloi !)

F51D  LDA Ø3ØØ    F578  LDA Ø3ØØ    prendre la réponse (Bit 3 )
F52Ø  AND #Ø8     F57B  AND #Ø8     et la tester.
F522  BNE F531    F57D  BNE F58C    sortir si la touche est pressée.
F524  DEX         F57F  DEX         passer à la ligne suivante,
F525  TXA         F58Ø  TXA
F526  AND #Ø7     F581  AND #Ø7     Modulo 8
F528  TAX         F583  TAX         et actualiser le pointeur.
F529  CMP Ø211    F584  CMP Ø211    si on a atteint la première ligne scrutée,
F52C  BNE F513    F587  BNE F56E
F52E  LDA #ØØ     F589  LDA #ØØ     on sort avec A=Ø,et surtout N=Ø.
F53Ø  RTS         F58B  RTS
F531  TXA         F58C  TXA         Une touche est pressée:
```

```
F532  ORA #80     F58D  ORA #80     on l'indique en mettant le Bit 7 à 1 (N=1).
F534  RTS         F58F  RTS
```

ROUTINE D'E/S AVEC LE PS6 8912

Entrée: A contient le numéro de registre et dans X la valeur à y placer.

Sortie: tous registres modifiés, sauf P

Remarque: il est impossible par cette routine de mettre le port en entrée, et encore moins de lire une valeur sur ce dernier, il est vrai que cela n'aurait aucun intérêt.

Sur la V1.0, on perd trop de temps entre le moment où on valide le numéro et où on place la valeur, ce qui le rend très pointilleux sur les 8912, et leur température ! La routine de la V1.1 est plus rapide, ce qui lui permet de n'avoir jamais aucun problème de clavier.
CE N'EST DONC PAS UN PROBLEME 'HARD' QUI EST RESPONSABLE DE TANT DE RETOURS D'ORIC-1 AU SAV !

```
F535  PHP         F590  PHP         Sauver P car on va toucher à I.
F536  SEI         F591  SEI         Interdire IRQ (problèmes de Timing).
F537  STA 030F    F592  STA 030F    Mettre le numéro de registre sur le port.
F53A  CMP #07     ..............    est-ce le registre d'autorisation ?
F53C  BNE F544    ..............    non,sauter
F53E  TXA         ..............    oui,prendre la valeur
F53F  ORA #40     ..............    et forcer Bit 6 (port A en sortie)
F541  JMP $F545   ..............    et continuer
F544  TXA         ..............    prendre valeur dans A
F545  PHA         ..............    et la sauver sur la pile
F546  LDX #0C     ..............    indiquer registre de contrôle (PCR)
F548  LDY #EE     ..............    préparer pour CA2 et CB2 à 1
F54A  LDA #11     ..............    faire ET avec #11 (sens de transition)
F54C  JSR $F56A   ..............    indiquer donc numéro de registre
F54F  LDY #CC     ..............    CA2 et CB2 à 0
F551  LDA #11     ..............
F553  JSR $F56A   ..............    valider le numéro de registre
F556  PLA         ..............    récupérer la donnée
F557  STA 030F    ..............    la mettre sur le port
F55A  LDY #EC     ..............    CA2 à 1,CB2 à 0
F55C  LDA #11     ..............
F55E  JSR $F56A   ..............    indiquer donnée
F561  LDY #CC     ..............
```

```
F563  LDA #11     ..............
F565  JSR $F56A   ..............  et valider la donnée
F568  PLP         ..............  récupérer P
F569  RTS         ..............
```

E/S avec le VIA 6522

Entrée: le numéro de registre du VIA est dans X (de Ø à #F donc), la routine fait un ET logique entre A et le registre concerné, puis un OU logique avec Y, et met le résultat dans le registre concerné.

Sortie: A contient la valeur finalement placée dans le registre.
X, Y et P inchangés.

Remarque: une bonne idée dans son principe, en fait très lourd à utiliser. D'ailleurs, a été abandonné sur la V1.1

```
F56A  PHP         ..............  Sauver l'indicateur d'IRQ surtout.
F56B  SEI         ..............  interdire les IRQ.
F56C  AND Ø300,X  ..............  faire un ET avec le registre,
F56F  STA Ø20F    ..............  sauver le résultat
F572  TYA         ..............  préparer A,
F573  ORA Ø20F    ..............  faire un OU avec la valeur précédente.
F576  STA Ø300,X  ..............  et écrire enfin le registre
F579  PLP         ..............  récupérer P.
F57A  RTS         ..............

..............  F595  TAY       numéro de registre dans Y
..............  F596  TXA       et donnée dans A
..............  F597  CPY #Ø7   est ce le registre d'autorisation ?
..............  F599  BNE F59D
..............  F59B  ORA #4Ø   oui,forcer le port A en sortie
..............  F59D  PHA       sauver la donnée sur la pile
..............  F59E  LDA Ø30C
..............  F5A1  ORA #EE   CA2 et CB2 á 1
..............  F5A3  STA Ø30C  indiquer numéro de registre
..............  F5A6  AND #11   garder les sens de transition
..............  F5A8  ORA #CC   CA2 et CB2 à Ø
..............  F5AA  STA Ø30C  validation du numéro de registre
..............  F5AD  TAX       on sauve la valeur de PCR dans X
..............  F5AE  PLA       on récupère la donnée
..............  F5AF  STA Ø30F  et la met sur le port
..............  F5B2  TXA
```

```
..... F5B3  ORA #EC     CA2 à 1 et CB2 à Ø
..... F5B5  STA Ø3ØC    indiquer donnée
..... F5B8  AND #11
..... F5BA  ORA #CC     CA2 et CB2 à Ø
..... F5BC  STA Ø3ØC    valider la donnée
..... F5BF  PLP         ;écupérer le registre d'etat.
..... F5CØ  RTS
```

M) GESTION DE L'IMPRIMANTE

1-Procédures standards

La gestion d'une imprimante est un exemple typique du Handshaking (échange avec 'poignée de main').

On met le caractère sur le port, on signale à l'imprimante que le caractère sur le port doit être imprimé (signal STROBE), et on attend que l'imprimante réponde qu'elle a bien imprimé le caractère (signal ACK) avant de recommencer.

Compte tenu de la configuration de l'Oric, voici un programme standard permettant d'envoyer A vers l'imprimante:

```
LDA #DONNEE
PHP             ;sauver I
SEI             ;éviter IRQ
STA VIADRA      ;mettre la donnée et effacer indicateur CA1 (pas d'ACK encore)
LDA VIADRB      ;prendre valeur courante du port B pour ne pas le perturber
AND #%111Ø1111 ;et forcer le STROBE (PB4) à Ø (Cf timing dans le chapitre
STA VIADRB      ;entrées/sorties)
ORA #%ØØØ1ØØØØ ;STROBE au repos (à 1) pour éviter la répétition du caractère
STA VIADRB
PLP             ;récupérer I
LDA #%ØØØØØØ1Ø ;masque pour indicateur CA1
B BIT VIAIFR    ;et attendre reception ACK reçu
BEQ B
```

Remarque: cette méthode suppose que l'imprimante soit prête. On peut au contraire commencer par attendre l'ACK, puis envoyer la donnée, ce qui aura pour avantage que l'Oric et l'imprimante travailleront simultanément et non alternativement.

Cette dernière méthode est cependant peu utilisable car l'imprimante n'envoie qu'un ACK, et la routine se bloquerait si elle ratait ce seul signal (encore que l'IFR garde l'indicateur. Une utilisation du signal BUSY de la norme Centronics aurait été plus pratique.

On peut générer des IRQ sur CA1 c'est à dire après chaque ACK: de quoi se faire un BUFFER pratique et pas cher !

2-Listing

ENVOYER A VERS L'IMPRIMANTE

Entrée: A contient le caractère à afficher

Sortie: (V1.1) Le caractère est affiché, X et Y n'ont pas été touchés.
(V1.Ø) Si le caractère a pu être envoyé, A=#ØØ et N=Ø, sinon, après 15 secondes, A=%FF et N=1

Bogue: V1.Ø Les interruptions restent autorisées. De telle sorte que s'il survient une IRQ entre le moment où on a placé la donnée sur le port, et le moment où le strobe est généré, l'imprimante affichera ce que l'IRQ aura mis sur le port, c'est à dire tout sauf ce qu'on voulait. (en fait, #7F, qui est la valeur de la dernière colonne du clavier balayée))

V1.1:Si l'imprimante n'est pas branchée, le programme boucle... Et faire une NMI ne sert à rien car celle-ci ne remet pas le drapeau d'imprimante, donc le Ready ira s'afficher sur l'imprimante, qui n'y est pas... Une seule solution: mettre le doigt sur le port imprimante, les tensions parasites ainsi crées feront croire qu'un ACK a été généré, mais c'est dangereux pour le VIA, à cause justement des tensions susnommées !

```
F57B  STA Ø3Ø1    ..............  mettre le caractère sur le port
F57E  LDX #ØØ     ..............  indexer registre #3ØØ (port B)
F58Ø  LDY #ØØ     ..............  OU avec Ø
F582  LDA #EF     ..............  et ET avec %111Ø1111
F584  JSR $F56A   ..............  c'est à dire générer un strobe
F587  LDY #1Ø     ..............  OU avec %ØØØ1ØØØØ
F589  LDA #FF     ..............  et avec %11111111
F58B  JSR $F56A   ..............  c'est à dire remettre PB4 à l'état haut
F58E  LDA #Ø2     ..............  indexer deuxième timer
F59Ø  LDX #Ø5     ..............
F592  LDY #DC     ..............  YX=#5DC=15ØØ,soit 15 secondes
F594  JSR $EBD3   ..............  et charger T2
F597  LDA Ø3ØD    ..............  prendre IFR (registre d'interruption)
F59A  AND #Ø2     ..............  tester le flag de l'ACK (CA1)
F59C  BNE F5AD    ..............  allumé:OK,on sort,N=Ø
F59E  LDA #Ø2     ..............  indexer T2
F5AØ  JSR $EBD6   ..............  prendre sa valeur dans XY
F5A3  TYA         ..............  si poids fort non nul,on continue
```

```
F5A4  BNE F597    ..............
F5A6  TXA         ..............  idem si poids faible non nul
F5A7  BNE F597    ..............
F5A9  LDA #FF     ..............  si T2=0,l'imprimante n'est pas branchée
F5AB  BMI F5B2    ..............  et on sort,N=1
F5AD  LDA 030D    ..............  inutile
F5B0  LDA #00     ..............  indiquer caractère envoyé (N=0)
F5B2  RTS         ..............
..............  F5C1  PHP        sauver autorisation d'IRQ
..............  F5C2  SEI        et les interdire..
..............  F5C3  STA 0301   mettre le caractère sur le port
..............  F5C6  LDA 0300   et générer un STROBE
..............  F5C9  AND #EF    c'est à dire PB4 à 0
..............  F5CB  STA 0300
..............  F5CE  LDA 0300   inutile
..............  F5D1  ORA #10    puis à 1
..............  F5D3  STA 0300
..............  F5D6  PLP        récupérer IRQ éventuelles
..............  F5D7  LDA 030D   et attendre le signal ACK
..............  F5DA  AND #02
..............  F5DC  BEQ F5D7
..............  F5DE  LDA 030D   inutile
..............  F5E1  RTS
```

TABLE DES DEPLACEMENTS POUR L'EXECUTION DES CARACTERES DE CONTROLES

```
F5B3 ....  BYT #6E,#6E,#6E,#6E
F5B7 ....  BYT #48,#6E,#4B,#6B
F5BB ....  BYT #00,#10,#1D,#0A
F5BF ....  BYT #32,#23,#29,#6E
F5C3 ....  BYT #4C,#4E,#6E,#4D
F5C7 ....  BYT #58,#6E,#6E,#6E
F5CB ....  BYT #6E,#6E,#4A,#4A
F5CF ....  BYT #6E,#49,#3F,#6E
.... F5E2  BYT #CF,#CF,#CF,#CF
.... F5E6  BYT #A3,#CF,#A6,#CC
.... F5EA  BYT #00,#27,#34,#0F
.... F5EE  BYT #66,#99,#60,#CF
.... F5F2  BYT #A7,#B3,#CF,#A8
.... F5F6  BYT #BE,#CF,#CF,#CF
.... F5FA  BYT #CF,#CF,#A5,#A5
.... F5FE  BYT #CF,#A4,#84,#CF
```

TRAITER LES CARACTERES DE CONTROLE

Entrée: A contient le code du caractère (seuls b0-b5 sont significatifs).

Sortie: tous registres affectés.

Principe: on calcule l'adresse de traitement, puis on efface le curseur, et on évite aussi son clignotement, en utilisant la manière forte (V1.0: interdire les IRQ), ou une meilleure méthode, en sauvant puis modifiant #026A. Ce qui oblige à sortir par #F6CC/#F6FE, pour tout remettre en place.

Les routines sont très différentes entre les deux versions, surtout en ce qui concerne la gestion des colonnes protégées, ignorée sur la V1.0. De plus, la V1.1 n'a pas de routine qui permet de calculer l'adresse de la Aème ligne, ce qui ajoute pas mal de lourdeurs lors des ajustements d'adresse de début de ligne...

Bogue: sur la V1.0, les IRQ sont automatiquement réautorisées à la sortie. Ce défaut n'apparait pas en temps normal, car la routine d'affichage d'un caractère qui appelle cette routine sauve P.

```
F5D3  AND #1F     F602  AND #1F     ramener à #00-#1F
F5D5  TAX         F604  TAX         comme index
F5D6  LDA F5B3,X  F605  LDA F5E2,X  prendre déplacement dans la table
F5D9  CLC         F608  CLC         et ajouter #F5F2/#F62F
F5DA  ADC #F2     F609  ADC #2F     pour obtenir l'adresse de traitement
F5DC  STA 0261    F60B  STA 0261    et sauver l'adresse poids faible
F5DF  LDA #00     F60E  LDA #00
F5E1  ADC #F5     F610  ADC #F6     calculer poids fort aussi
F5E3  STA 0262    F612  STA 0262    et le sauver pour indirection
F5E6  SEI         ..... ..........  interdire les IRQ
..... ..........  F615  LDA 026A   sauver le registre d'état
..... ..........  F618  PHA
..... ..........  F619  AND #FE    faire curseur OFF
..... ..........  F61B  STA 026A   et replacer
..... ..........  F61E  PLA         récupérer le registre
..... ..........  F61F  AND #01     isoler le bit du curseur
..... ..........  F621  STA 0251    et le sauver
F5E7  LDA #00     F624  LDA #00
F5E9  JSR $F7CB   F626  JSR $F801   effacer le curseur
F5EC  SEC         F629  SEC         C=1 (préparer pour décalages)
F5ED  LDA #00     F62A  LDA #00     A=0, même raison
F5EF  JMP (0261)  F62C  JMP (0261)  et exécuter le caractère
```

Curseur gauche (BS);Code ASCII=#08

```
F5F2  DEC Ø269    ..............  décrémenter la colonne
F5F5  BPL F621    ..............  c'est tout si sur la même ligne
F5F7  LDA #27     ..............  sinon, mettre curseur dernière colonne
F5F9  STA Ø269    ..............  et faire curseur haut
..............  F62F  DEC Ø269   décrémenter la colonne
..............  F632  BMI F639   sauter si changer de ligne
..............  F634  JSR $F7D7  tester colonne protégée
..............  F637  BNE F679   non, on sort
..............  F639  LDA #27    placer curseur à la dernière colonne
..............  F63B  STA Ø269
```

Curseur haut (VT);Code ASCII=#ØB

```
F5FC  DEC Ø268    ..............  décrémenter la ligne
F5FF  JMP $F67D   ..............  ajuster adresse, scroll éventuel

..............  F63E  LDA Ø268   prendre ligne
..............  F641  CMP #Ø1    est-ce la première ?
..............  F643  BEQ F679   oui, on sort (pas de scroll vers le bas)
..............  F645  DEC Ø268   non, on décrémente la ligne
..............  F648  SEC        et on calcule la nouvelle
..............  F649  LDA 12     adresse de début de ligne
..............  F64B  SBC #28    ce qui revient à enlever 4Ø
..............  F64D  STA 12      à l'adresse courante.
..............  F64F  BCS F653   poids fort aussi...
..............  F651  DEC 13
..............  F653  JMP $F6FE  et finir
```

Curseur droit (HT);Code ASCII=#Ø9

```
F602  INC Ø269    ..............  incrémenter colonne
F605  LDX #27     ..............  et tester si on doit passer à la ligne
F607  CPX Ø269    ..............  suivante
F60A  BPL F621    ..............  non, c'est tout
F60C  STA Ø269    ..............  colonne=Ø (A vaut Ø depuis le début)
..............  F656  INC Ø269   incrémenter la colonne
..............  F659  LDX #27    et tester si il faut passer à la
..............  F65B  CPX Ø269   ligne suivante
..............  F65E  BPL F679   non, on sort
..............  F660  JSR $F70D  oui, placer le curseur au début de la ligne
```

Curseur bas (LF);Code ASCII=#ØA

```
F60F  INC Ø268    ..............  incrémenter la ligne
```

```
F612  JMP $F67D  ..............  ajuster adresse,scroll éventuel

..............  F663  LDA Ø268   prendre ligne courante
..............  F666  CMP Ø27E   et comparer à maxi permise
..............  F669  BEQ F67C   oui, faire le scroll
..............  F66B  INC Ø268   :on, ligne suivante
..............  F66E  CLC        et ajuster l'adresse de début
..............  F66F  LDA 12     en ajoutant 4Ø à l'adresse courante
..............  F671  ADC #28
..............  F673  STA 12
..............  F675  BCC F679   sans oublier le poids fort
..............  F677  INC 13
..............  F679  JMP $F6FE  et finir
..............  F67C  JSR $F35D  sauver #1Ø-#11
..............  F67F  LDX #Ø6    et décaler la table des paramètres
..............  F681  LDA Ø277,X
..............  F684  STA ØB,X   vers la zone #ØC-#11
..............  F686  DEX
..............  F687  BNE F681
..............  F689  JSR $EDC4  effectuer le décalage (scroll)
..............  F68C  JSR $F36E  récupérer #1Ø-#11

Retour chariot (CR);Code ASCII=#ØD

F615  STA Ø269   ..............
F618  JMP $F6CC  ..............  et finir

Effacement ligne (SO);Code ASCII=#ØE

F61B  LDA Ø268   ..............  prendre ligne
F61E  JSR $F6D3  ..............  effacer la ligne dans A
F621  JMP $F6CC  ..............  et finir
..............  F68F  JSR $F71A  effacer la ligne (selon #12-#13)
..............  F692  JMP $F6FE  et finir

Effacer l'écran (FF);Code ASCII=#ØC

F624  TAX        ..............  X=Ø
F625  INX        ..............  passer à la ligne suivante
F626  TXA        ..............  ligne dans A
F627  JSR $F6D3  ..............  effacer la Aème ligne
F62A  CPX Ø26F   ..............  et continuer autant que de lignes
F62D  BNE F625   ..............
..............  F695  LDX Ø27E   prendre le nombre de ligne de l'écran
```

```
..............  F698  LDA Ø27A
..............  F69B  STA 12
..............  F69D  LDA Ø27B    initialiser #12-#13 sur la première
..............  F6AØ  STA 13      ligne à effacer
..............  F6A2  JSR $F71A   effacer la ligne courante
..............  F6A5  CLC
..............  F6A6  LDA 12
..............  F6A8  ADC #28     passer à la ligne suivante
..............  F6AA  STA 12
..............  F6AC  BCC F6BØ
..............  F6AE  INC 13
..............  F6BØ  DEX         s'il en reste...
..............  F6B1  BNE F6A2

Curseur HOME (RS);Code ASCII=#1E

F62F  LDA #ØØ     ..............
F631  STA Ø269    ..............  colonne=Ø
F634  STA Ø268    ..............  ligne aussi
F637  BEQ F6ØF    ..............  inconditionnel: faire un curseur bas

..............  F6B3  JSR $F7ØD   placer le curseur au début de la ligne
..............  F6B6  LDA #Ø1     et initialiser
..............  F6B8  STA Ø268    le numéro de ligne
..............  F6BB  LDA Ø27A
..............  F6BE  STA 12      ainsi que son adresse
..............  F6CØ  LDA Ø27B
..............  F6C3  STA 13
..............  F6C5  JMP $F6FE   et finir

Retour chariot (CR);Code ASCII=#ØD

..............  F6C8  JSR $F7ØD   placer le curseur au début de la ligne
..............  F6CB  STX Ø253    et sauver première position valide
..............  F6CE  JMP $F6FE   puis finir


Inutlisé

F639  ROL A       F6D1  ROL A       placer 1 dans b7

Double hauteur (EOT);Code ASCII=#Ø4

F63A  ROL A       F6D2  ROL A       placer 1 dans b6
```

```
Protéction colonnes (GS);Code ASCII=#1D

F63B  ROL A       F6D3  ROL A       placer 1 dans b5

ESCAPE (ESC);Code ASCII=#1B ou #1A

F63C  ROL A       F6D4  ROL A       placer 1 dans b4

Son clavier (ACK);Code ASCII=#06

F63D  ROL A       F6D5  ROL A       placer 1 dans b3

Inutilisé (DLE);Code ASCII=#10

F63E  ROL A       F6D6  ROL A       placer 1 dans b2

Affichage (DC3);Code ASCII=#13

F63F  ROL A       F6D7  ROL A       placer 1 dans b1

Clignotement (DC1);Code ASCII=#11  (Sauf V1.1)

F640  ROL A       F6D8  ROL A       placer 1 dans b0
F641  EOR 026A    F6D9  EOR 026A    iverser le drapeau correspondant
F644  STA 026A    F6DC  STA 026A    et le resauver
F647  JMP $F6CC   F6DF  JMP $F6FE   et finir

Clignotement (DC1);Code ASCII=#11

.............. F6E2  LDA 0251     prendre couleur (sauvegardée)
.............. F6E5  EOR #01      et l'inverser
.............. F6E7  STA 0251     puis resauver
.............. F6EA  JMP $F6FE    et finir

Maj/Min (DC4);Code ASCII=#14

F64A  LDA 020C    F6ED  LDA 020C    prendre masque courant
F64D  EOR #80     F6F0  EOR #80     inverser drapeau
F64F  STA 020C    F6F2  STA 020C    et resauver
F652  LDA 021F    ..............
F655  BNE F65A    ..............    sortir si mode HIRES
F657  JSR $F729   F6F5  JSR $F75A   allumer ou éteindre CAPS
F65A  JMP $F6CC   F6F8  JMP $F6FE   et finir
```

```
PING (BEL);Code ASCII=#07

F65D  JSR $FA85   F6FB  JSR $FA9F   faire PING
F660  CLC         ..............
F661  BCC F6CC    ..............   et finir (pourquoi pas JMP ?)

Finir de traiter un caractère

..............    F6FE  LDA #26A    prendre registre d'état
..............    F701  ORA #251    et remettre couleur du curseur
..............    F704  STA #26A    et resauver
..............    F707  LDA #01     et enfin réafficher le curseur
..............    F709  JSR $F801
..............    F70C  RTS         JMP #F801 aurait été bien aussi !

Placer curseur au début de la ligne

..............    F70D  LDX #00     indexer première colonne
..............    F70F  JSR $F7DE   et tester si protégée
..............    F712  BNE F716    non, on sort
..............    F714  INX         oui, première colonne valide=2
..............    F715  INX
..............    F716  STX #269    dans numéro de colonne
..............    F719  RTS

Messages

Les auteurs avaient surement prévu que Ctrl P fasse office de bascule
imprimante, ainsi que l'annonçait le manuel de l'Oric-1. Il aurait suffi de pas
grand chose, même le message sur la ligne de STATUS était prêt !

F663 ....  BYT #07,"PRINT",#00
F66A ....  BYT #07,"     ",#00
F671 ....  BYT #07,"CAPS",#00
F677 ....  BYT #07,"    ",#00

Déplacements verticaux,suite.

Principe: c'est cette routine qui effectue les scrolls. Scroller l'écran
revient à le descendre d'une ligne ou à le remonter d'une ligne, et ensuite
effacer la ligne du curseur, pour ne pas dédoubler cette dernière. La lenteur
vient de la routine #EBD9.
```

```
F67D  LDA Ø268    ..............  prendre nouvelle ligne
F68Ø  JSR $F6F1   ..............  et calculer dans AY son adresse
F683  STA 12      ..............
F685  STY 13      ..............  et sauver comme adresse de début de ligne
F687  LDX Ø26F    ..............  prendre nombre de lignes
F68A  DEX         ..............  -1
F68B  TXA         ..............  dans A pour calcul
F68C  JSR $F7ØØ   ..............  calculer 4Ø‡A
F68F  STA Ø2Ø4    ..............
F692  STY Ø2Ø5    ..............  et sauver le résultat
F695  LDX Ø268    ..............  prendre nouvelle colonne demandée
F698  BEQ F6A9    ..............  si Ø,scroll vers le bas
F69A  DEX         ..............  ajuster
F69B  CPX Ø26F    ..............
F69E  BNE F6CC    ..............  si pas haut de la page, on sort
F6AØ  DEC Ø268    ..............  scroll vers le haut: ajuster ligne
F6A3  LDA #Ø1     ..............  ligne cible=1
F6A5  LDX #Ø2     ..............  ligne source=2
F6A7  BNE F6BØ    ..............
F6A9  INC Ø268    ..............  scroll vers le bas: ajuster ligne
F6AC  LDA #Ø2     ..............  ligne cible=2
F6AE  LDX #Ø1     ..............  ligne source=1
F6BØ  JSR $F6F1   ..............  calculer adresse ligne cible
F6B3  STA Ø2Ø2    ..............
F6B6  STY Ø2Ø3    ..............  et la sauver
F6B9  TXA         ..............  ligne origine dans A
F6BA  JSR $F6F1   ..............  on calcule son adresse
F6BD  STA Ø2ØØ    ..............
F6CØ  STY Ø2Ø1    ..............  et on sauve
F6C3  JSR $EBD9   ..............  effectuer le décalage
F6C6  LDA Ø268    ..............  prendre ligne courante
F6C9  JSR $F6D3   ..............  et l'effacer
```

Finir de traiter un caractère

```
F6CC  LDA #Ø1     ..............  afficher le curseur
F6CE  JSR $F7CB   ..............
F6D1  CLI         ..............  et autoriser IRQ
F6D2  RTS         ..............
```

Effacer Aème ligne

```
F6D3  JSR $F6F1   ..............  calculer adresse ligne courante
F6D6  STA 12      ..............
```

```
F6D8  STY 13      ..............  et la sauver
```

Effacer ligne courante

```
F6DA  LDY #27     F71A  LDY #27     indexer dernière colonne
F6DC  LDA #' '    F71C  LDA #' '    prendre code de l'espace
F6DE  STA (12),Y  F71E  STA (12),Y  et envoyer à l'écran
F6E0  DEY         F720  DEY         colonne précedente
F6E1  BPL F6DE    F721  BPL F71E    jusqu'à colonne 0
F6E3  LDY #00     F723  LDY #00     INY aurait marché !
F6E5  LDA 026B    F725  LDA 026B    prendre attribut premiére colonne
F6E8  STA (12),Y  F728  STA (12),Y  et sauver
F6EA  LDA 026C    F72A  LDA 026C    prendre attribut deuxième colonne
F6ED  INY         F72D  INY
F6EE  STA (12),Y  F72E  STA (12),Y  et sauver
F6F0  RTS         F730  RTS
```

Calculer adresse Aème ligne (TEXT ou HIRES)

Entrée: A contient le numéro de la ligne.

Sortie: AY contient l'adresse de la ligne, X est inchangé.

```
F6F1  JSR $F700   ..............  AY=40*A
F6F4  CLC         ..............
F6F5  ADC 026D    ..............
F6F8  PHA         ..............  ajouter poids faible début écran
F6F9  TYA         ..............
F6FA  ADC 026E    ..............
F6FD  TAY         ..............  idem poids fort
F6FE  PLA         ..............
F6FF  RTS         ..............
```

AY=40*A

Entrée: A contient la valeur voulue...

Sortie: AY contient 40*A (#263=Y et #264=A original), X est inchangé.

```
F700  LDY #00     F731  LDY #00     initialiser le résultat poids fort
F702  STY 0263    F733  STY 0263    à 0
F705  STA 0264    F736  STA 0264    sauver A pour addition
```

```
F708  ASL A       F739  ASL A
F709  ROL 0263    F73A  ROL 0263    X2
F70C  ASL A       F73D  ASL A
F70D  ROL 0263    F73E  ROL 0263    X4
F710  CLC         F741  CLC
F711  ADC 0264    F742  ADC 0264
F714  BCC F719    F745  BCC F74A
F716  INC 0263    F747  INC 0263    X5 (X4+original)
F719  ASL A       F74A  ASL A
F71A  ROL 0263    F74B  ROL 0263    X10
F71D  ASL A       F74E  ASL A
F71E  ROL 0263    F74F  ROL 0263    X20
F721  ASL A       F752  ASL A
F722  ROL 0263    F753  ROL 0263    X40
F725  LDY 0263    F756  LDY 0263    récupérer le poids fort
F728  RTS         F759  RTS
```

Ajuster CAPS

```
F729  LDA 020C    F75A  LDA 020C    prendre masque
F72C  BPL F735    F75D  BPL F766    si minuscule effacer
F72E  LDA #71     F75F  LDA #70
F730  LDY #F6     F761  LDY #F7     indexer le message 'CAPS'
F732  JMP $F739   F763  JMP $F76A   et afficher (BMI aurait été bien)
F735  LDA #77     F766  LDA #76
F737  LDY #F6     F768  LDY #F7     indexer le message '    '
F739  LDX #23     F76A  LDX #23     indiquer afficher à la colonne 35
F73B  JSR $F82F   F76C  JSR $F865   afficher
F73E  RTS         F76F  RTS         (JMP $F82F/$F865 aurait marché !)
```

Messages

```
.... F770  BYT #07,"CAPS",#00
.... F776  BYT #07,"    ",#00
```

AFFICHER X A L'ECRAN

Entrée: X contient le code ASCII du caractère à afficher

Sortie: A,X,Y,C,V,I,B inchangés. N et Z placés selon A

Principe: on teste d'abord le code du Ctrl-S, pour pouvoir l'annuler,et aussi

quelques autres qui ne seront donc pas interdits par le Ctrl-S. Le reste est trivial à suivre. Quelques complications sur la V1.Ø pour ignorer les colonnes protégées.

```
F73F  PHA          F77C  PHA          sauver A
F74Ø  PHP          F77D  PHP          sauver P
F741  TYA          F77E  TYA
F742  PHA          F77F  PHA          sauver Y
F743  TXA          F78Ø  TXA          et code dans A
F744  PHA          F781  PHA          et enfin X
F745  CLD          F782  CLD          imposer mode décimal (?)
F746  CMP #2Ø      ..... ..... est-ce un caractère de controle ?
F748  BCC F795     ..... ..... oui, le traiter et c'est tout
..... ..... F783  CPX #13    est-ce Ctrl-S ?
..... ..... F785  BEQ F7CD   oui, exécution
..... ..... F787  CPX #14    Ctrl-T ?
..... ..... F789  BEQ F7CD   oui, exécution
..... ..... F78B  CPX #Ø6    est-ce Ctrl-F ?
..... ..... F78D  BEQ F7CD   oui, exécution
F74A  LDA Ø26A     F78F  LDA Ø26A     prendre registre d'état
F74D  AND #Ø2      F8Ø2  AND #Ø2      et tester le drapeau d'affichage
F74F  BEQ F798     F8Ø4  BEQ F7DØ     si OFF, c'est fini...
..... ..... F796  TXA        prendre code dans A
..... ..... F797  CMP #2Ø    et tester si caractère de controle
..... ..... F799  BCC F7CD   oui, effectuer et finir
F751  LDA Ø26A     F79B  LDA Ø26A     tester le drapeau ESC
F754  AND #1Ø      F79E  AND #1Ø
F756  BEQ F76B     F7AØ  BEQ F7B5     non, sauter
F758  TXA          F7A2  TXA          oui, code dans A
F759  SEC          F7A3  SEC
F75A  SBC #4Ø      F7A4  SBC #4Ø      ne doit pas être inférieur à '@'
F75C  BMI F767     F7A6  BMI F7B1     si oui, afficher un espace
F75E  AND #1F      F7A8  AND #1F      ramener à #ØØ-#1F (attributs vidéo)
F76Ø  JSR $F7AC    F7AA  JSR $F7E4    afficher
F763  LDA #1B      F7AD  LDA #1B      prendre code pour 'ESC'
F765  BNE F795     F7AF  BNE F7CD     inconditionnel: remettre le drapeau ESC
F767  LDA #2Ø      F7B1  LDA #2Ø      afficher un espace
F769  BPL F76Ø     F7B3  BPL F7AA     inconditionnel
F76B  CPX #7F      F7B5  CPX #7F      est-ce DEL ?
F76D  BEQ F784     F7B7  BEQ F7C1     oui, on saute
F76F  JSR $F79F    ..... ..... tester colonne protégée
F772  BEQ F77C     ..... ..... oui, on se décale
F774  PLA          F7B9  PLA          récupérer le code
F775  PHA          F7BA  PHA          sans toucher à la pile
```

```
F776  JSR $F7AC   F7BB  JSR $F7E4   afficher le code
F779  JMP $F798   F7BE  JMP $F7D0   et sortir
F77C  LDA #09     ..............    prendre code de 'curseur droite'
F77E  JSR $F5D3   ..............    et l'effectuer
F781  JMP $F76F   ..............    puis recommencer
F784  LDA #08     F7C1  LDA #$08    DEL: déplacer le curseur vers la gauche
F786  JSR $F5D3   F7C3  JSR $F602
F789  JSR $F79F   ..............    tester colonne protégée
F78C  BEQ F784    ..............    et recommencer si colonne protégée
F78E  LDA #' '    F7C6  LDA #' '    afficher un espace pour effacer
F790  JSR $F7AC   F7C8  JSR $F7E4   le caractère sous le curseur
F793  LDA #08     F7CB  LDA #08     et revenir sur le caractère
F795  JSR $F5D3   F7CD  JSR $F602   afficher un caractère de controle
F798  PLA         F7D0  PLA         récupérer X
F799  TAX         F7D1  TAX
F79A  PLA         F7D2  PLA         puis Y
F79B  TAY         F7D3  TAY
F79C  PLP         F7D4  PLP         puis P
F79D  PLA         F7D5  PLA         et A...
F79E  RTS         F7D6  RTS
```

TESTER COLONNES PROTEGEES

Entrée: rien

Sortie: Z=1 si le curseur est dans la zone protégée (colonne 0 et 1) et que le drapeau indique que la zone est effectivement protégée.
X et Y inchangés.

```
F79F  LDA 0269    F7D7  LDA 0269    prendre position courante du curseur
F7A2  AND #FE     F7DA  AND #FE     et isoler le poids fort
F7A4  BNE F7AB    F7DC  BNE F7E3    poids fort non nul, colonne ) 1, Z=0.
F7A6  LDA 026A    F7DE  LDA 026A    si colonne protégée (0 ou 1)
F7A9  AND #20     F7E1  AND #20     positionner Z selon état du drapeau
F7AB  RTS         F7E3  RTS
```

AFFICHER A SELON #12-#13 ET #269

Entrée: A contient le code du caractère à afficher

Sortie: A et X ne sont pas affectés.

Bogue: s'il y a double hauteur, et que le curseur est sur la dernière ligne, la zone #BFE0-#C007 va être affectée. Dangereux si on y place des extensions comme le préconise le manuel...

```
F7AC  PHA          F7E4  PHA          sauver le code
F7AD  LDY #269     F7E5  LDY #269     prendre abscisse du curseur
F7B0  STA (12),Y   F7E8  STA (12),Y   et l'envoyer à l'écran
F7B2  LDA #26A     ..............     tester la drapeau de double hauteur
F7B5  AND #40      ..............
F7B7  BEQ F7C4     ..............     non, c'est fini
..............     F7EA  BIT #26A     c'est tellement plus simple...
..............     F7ED  BVC F7FA
F7B9  LDA #269     F7EF  LDA #269     (pourquoi pas TYA ?) ajuster l'index
F7BC  CLC          F7F2  CLC          sur la ligne suivante
F7BD  ADC #28      F7F3  ADC #28
F7BF  TAY          F7F5  TAY          à nouveau dans l'index
F7C0  PLA          F7F6  PLA          récupérer le code
F7C1  PHA          F7F7  PHA          tout en le laissant sur la pile
F7C2  STA (12),Y   F7F8  STA (12),Y   et l'envoyer à l'écran
F7C4  LDA #09      F7FA  LDA #09      déplacer le curseur vers la droite
F7C6  JSR $F5D3    F7FC  JSR $F602    pour ajuster sur la suite
F7C9  PLA          F7FF  PLA          et récupérer le code
F7CA  RTS          F800  RTS
```

EFFACER OU ETEINDRE LE CURSEUR

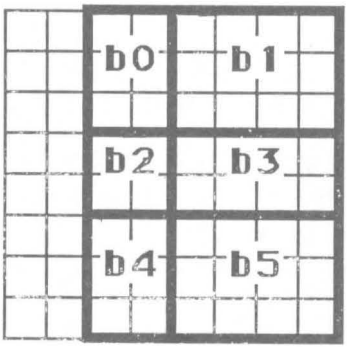
Entrée: A=0 ou 1 pour éteindre ou allumer le curseur (c'est à dire vidéo inverse ou non)

Programmation: une variable de travail aurait pu être évitée,en procédant comme suit: AND #26A:LDY #269:LSR A:PHP:LDA ($12),Y:ASL:PLP:ROR:STA ($12),Y

```
F7CB  AND #26A     F801  AND #26A     éteindre si curseur OFF
F7CE  LSR A        F804  LSR A        couleur dans C
F7CF  ROR A        F805  ROR A        et dans b7
F7D0  STA #265     F806  STA #265     on sauve
F7D3  LDY #269     F809  LDY #269     on prend l'abscisse
F7D6  LDA (12),Y   F80C  LDA (12),Y   et le caractère sous le curseur
F7D8  AND #7F      F80E  AND #7F      on élimine b7
F7DA  ORA #265     F810  ORA #265     pour le mettre á la couleur demandée
F7DD  STA (12),Y   F813  STA (12),Y   et on renvoie le code
F7DF  RTS          F815  RTS
```

GENERER LES CARACTERES ALTERNES

Principe: la structure des caractères alternés n'est pas aléatoire. Leur code s'obtient en les considérant comme des blocs de 6 bits, qui prennent toutes les valeurs possibles. Ces bits sont en fait des blocs dans la matrice, placés comme ci-dessous.



**** FIGURE F7-4 ****

Si on considère le motif comme équivalent à b5 b4 b3 b2 b1 bØ, il va falloir décrire toutes les valeurs pour chaque octet. On va en fait procéder par groupe de deux bits: bØ-b1,b2-b3 et enfin b4-b5. Chaque couple de deux bits donne un motif parmi 4. De plus, selon la position, le motif sera répété 2 ou trois fois, conformément au schéma.

Programmation: le principe est bien trouvé, mais la programmation laisse à désirer,on peut réduire presque de moitié cette routine...

```
F7EØ  LDA #ØØ     F816  LDA #ØØ
F7E2  STA ØC      F818  STA ØC
F7E4  LDA #B9     F81A  LDA #B9     initialiser pointeur de caractère
F7E6  STA ØD      F81C  STA ØD      à #B9ØØ, début de la table
F7E8  LDA #ØØ     F81E  LDA #ØØ     indiquer poids faible=Ø
F7EA  JSR $F7F7   F82Ø  JSR $F82D   et calculer 32 caractère
F7ED  LDY #BA     F823  LDY #BA     indexer maintenant #BAØØ
F7EF  STY ØD      F825  STY ØD      (INC #BA aurait marché !)
F7F1  LDA #2Ø     F827  LDA #2Ø     intialiser pour les 32 derniers (b5=1)
F7F3  JSR $F7F7   F829  JSR $F82D   et les calculer
F7F6  RTS         F82C  RTS

Calculer 32 caractères

F7F7  LDY #ØØ     F82D  LDY #ØØ     initialiser index
```

```
F7F9  PHA          F82F  PHA          sauver code en cours
F7FA  JSR $F81E    F830  JSR $F854   et calculer motif selon b0-b1
F7FD  STA (0C),Y   F833  STA (0C),Y  et une troisième fois
F7FF  INY          F835  INY
F800  PLA          F836  PLA          récupérer le code
F801  PHA          F837  PHA
F802  JSR $F81C    F838  JSR $F852   et calculer motif central selon b2-b3
F805  PLA          F83B  PLA          récupérer le code
F806  PHA          F83C  PHA
F807  JSR $F81A    F83D  JSR $F850   et calculer motif selon b5-b6
F80A  STA (0C),Y   F840  STA (0C),Y  et une troisième fois
F80C  INY          F842  INY
F80D  CPY #00      F843  CPY #00      inutile !!!. tester si une page (32*8=256)
F80F  BEQ F818     F845  BEQ F84E     ,c'est à dire 32 caractères a été décrite
F811  PLA          F847  PLA          non, on incrémente le pointeur de caractère
F812  CLC          F848  CLC
F813  ADC #01      F849  ADC #01
F815  JMP $F7F9    F84B  JMP $F82F    (BCC aurait marché) et recommencer
F818  PLA          F84E  PLA          sortir:ajuster la pile
F819  RTS          F84F  RTS
F81A  LSR A        F850  LSR A        décaler b4-b5 dans b0-b1
F81B  LSR A        F851  LSR A
F81C  LSR A        F852  LSR A        décaler b2-b3 dans b0-b1
F81D  LSR A        F853  LSR A
F81E  AND #03      F854  AND #03      garder seulement b0-b1
F820  TAX          F856  TAX          comme index
F821  LDA F82B,X   F857  LDA F861,X   prendre motif correspondant
F824  STA (0C),Y   F85A  STA (0C),Y   le sauver
F826  INY          F85C  INY
F827  STA (0C),Y   F85D  STA (0C),Y   au moins deux fois,c'est le minimum
F829  INY          F85F  INY
F82A  RTS          F860  RTS

Table des valeurs

F82B F861  BYT #00,#F0,#0F,#FF
```

ECRIRE SUR LA LIGNE 0

Entrée: AY pointe sur la chaine à afficher (elle doit se terminer par un 0), et X indique la colonne où doit commencer l'affichage.

Bogue: sur la VI.0,l'affichage est fait même en mode HIRES, ce qui n'est pas du

plus bel effet.

```
F82F  STA ØC      F865  STA ØC
F831  STY ØD      F867  STY ØD      sauver adresse du message
.............     F869  LDA Ø21F    prendre indicateur de mode
.............     F86C  BNE F87B    et sortir si mode HIRES
F833  LDY #ØØ     F86E  LDY #ØØ     préparer l'index
F835  LDA (ØC),Y  F87Ø  LDA (ØC),Y  prendre caractère de la chaine
F837  BEQ F84Ø    F872  BEQ F87B    et sortir si c'est la fin
F839  STA BB8Ø,X  F874  STA BB8Ø,X  l'afficher
F83C  INX         F877  INX         suivant à l'écran
F83D  INY         F878  INY         suivant dans la chaine
F83E  BNE F835    F879  BNE F87Ø    et on continue...
F84Ø  RTS         F87B  RTS
```

DONNEES POUR VECTEURS SYSTEMES

```
F841  JMP $EC03   .............  IRQ
F844  JMP $F43Ø   .............  NMI
F847  ORA (ØØ,X)  .............  ???
F849  RTI         .............  retour IRQ
.............     F87C  JMP $F77C  afficher un caractère
.............     F87F  JMP $EB78  prendre une touche au clavier
.............     F882  JMP $F5C1  envoyer un caractèrer à l'imprimante
.............     F885  JMP $F865  afficher ligne Ø
.............     F888  JMP $EE22  IRQ
.............     F88B  JMP $F8B2  NMI
.............     F88E  RTI        retour IRQ
```

RESET SYSTEME (COLD START)

```
F84A  LDX #FF     F88F  LDX #FF     initialiser le pointeur de pile
F84C  TXS         F891  TXS         au sommet
F84D  CLI         F892  CLI         autoriser IRQ
F84E  CLD         F893  CLD         et interdire mode décimal
F84F  LDX #Ø8     F894  LDX #12     transférer la table des vecteurs systèmes
F851  LDA F841,X  F896  LDA F87C,X
F854  STA Ø228,X  F899  STA Ø238,X  en mémoire vive
F857  DEX         F89C  DEX
F858  BPL F851    F89D  BPL F896
.............     F89F  LDA #2Ø
```

```
.............. F8A1  STA Ø24E    répétition normale=32
.............. F8A4  LDA #Ø4
.............. F8A6  STA Ø24F    rapide=4
F85A  JSR $F9CØ  F8A9  JSR $FA14  tester la mémoire
F85D  PHP        .............. sauver résultat du test
F85E  JSR $F888  F8AC  JSR $F8B8  faire NMI (écran,VIA...)
F861  PLP        .............. reprendre résultat du test
F862  BEQ F871   .............. sauter si OK
F864  LDY #ØØ    ..............
F866  LDX F874,Y .............. sinon, afficher
F869  JSR $F73F  .............. le message 'MEMORY ERROR'
F86C  INY        ..............
F86D  CPY #ØE    .............. ,qui a 14 caractères
F86F  BNE F866   ..............
F871  JMP $CØØØ  F8AF  JMP $ECCC  faire COLD START du BASIC
```

NMI SYSTEME (WARM START)

```
F882  JSR $F886  F8B2  JSR $F8B8  configurer écran, via etc...
F885  JMP $CØØ3  F8B5  JMP $C471  et faire WARM START BASIC
```

CONFIGURER LE SYSTEME

Action: configurer le système des entrées/sorties (PSG 8912, VIA etc...), ainsi que le terminal écran.

```
F888  JSR $F96Ø  F8B8  JSR $F9AA  configurer VIA,IRQ inhibées
F88B  LDA #Ø7    F8BB  LDA #Ø7    indexer registre d'autorisation du GI 8912
F88D  LDX #4Ø    F8BD  LDX #4Ø    ouvrir les canaux (!) et port A en sortie
F88F  JSR $F4Ø6  F8BF  JSR $F59Ø
F892  JSR $EBDØ  F8C2  JSR $EDEØ  autoriser IRQ, placer TIMER etc...
F895  JSR $F8D1  F8C5  JSR $F9ØE  placer quelques registres d'état affichage
.............. F8C8  LDA #FF    indiquer mode majuscule
.............. F8CA  STA Ø2ØC
F898  JSR $F97F  F9CD  JSR $F9C9  faire TEXT
F89B  LDX #Ø5    F8DØ  LDX #Ø5    indexer 1er groupe de valeurs
F89D  JSR $F93E  F8D2  JSR $F982  et décaler caractères ROM --) RAM
F8AØ  JSR $F4ØC  F8D5  JSR $F816  générer le deuxième jeu de caractère
F8A3  LDA #FF    .............. indiquer mode majuscule
F8A5  STA Ø2ØC   ..............
.............. F8D8  JSR $F75A  afficher CAPS
F8A8  RTS        F8DB  RTS
```

HIRES:CREER FENETRE TEXT

```
F8A9  PHA          F8DC  PHA          sauver A
F8AA  TXA          F8DD  TXA
F8AB  PHA          F8DE  PHA          et X
F8AC  LDA #00      ..................
F8AE  STA 0269     ..................  curseur sur colonne 0
F8B1  LDA #01      F8DF  LDA #01
F8B3  STA 021F     F8E1  STA 021F     indiquer mode HIRES
F8B6  STA 0268     ..................  et ligne 1
F8B9  LDA #40      ..................  placer adresse ligne 0 en #BF40
F8BB  STA 026D     ..................
F8BE  LDA #BF      ..................
F8C0  STA 026E     ..................
..................  F8E4  LDA #BF
..................  F8E6  STA 027B
..................  F8E9  STA 0279
..................  F8EC  LDA #68      adresse ligne 1=#BF68
..................  F8EE  STA 027A
..................  F8F1  LDA #90      adresse ligne 2=#BF90
..................  F8F3  STA 0278
F8C3  LDA #03      F8F6  LDA #03
F8C5  STA 026F     F8F8  STA 027E     indiquer 3 lignes d'écran
..................  F8FB  LDA #00
..................  F8FD  STA 027D     nombre de caractère à scroller= #0050
..................  F900  LDA #50      c'est à dire deux lignes
..................  F902  STA 027C
F8C8  LDX #0C      F905  LDX #0C
F8CA  JSR $F409    F907  JSR $0238    effacer l'écran
F8CD  PLA          F90A  PLA
F8CE  TAX          F90B  TAX
F8CF  PLA          F90C  PLA          et récupérer les registres
F8D0  RTS          F90D  RTS
```

INITIALISER COULEUR ECRAN

Entrée: rien

Sortie: A,X et Y inchangés

```
F8D1  PHA          F90E  PHA          sauver A
F8D2  LDA #03      F90F  LDA #03      mettre affichage, curseur, son en service
F8D4  STA 026A     F911  STA 026A     et mode 40 colonne,ESC hors service
```

```
F8D7  LDA #ØØ     F914  LDA #ØØ     couleur d'encre =noir
F8D9  STA Ø26C    F916  STA Ø26C
F8DC  LDA #17     F919  LDA #17     couleur de fond =blanc
F8DE  STA Ø26B    F91B  STA Ø26B
F8E1  PLA         F91E  PLA
F8E2  RTS         F91F  RTS
```

CRRER L'ECRAN HIRES

```
F8E3  PHA         F92Ø  PHA         sauver A
F8E4  LDA Ø21F    F921  LDA Ø21F    déjà HIRES ?
F8E7  BNE F8EE    F924  BNE F92B    oui,
on saute
F8E9  LDX #ØB     F926  LDX #ØB     non, on décale les caractères
F8EB  JSR $F93E   F928  JSR $F982
F8EE  LDA #FE     F92B  LDA #FE     on interdit le clignotement
F8FØ  AND Ø26A    F92D  AND Ø26A    du curseur
F8F3  STA Ø26A    F93Ø  STA Ø26A
F8F6  LDA #1E     F933  LDA #1E     mettre attribut HIRES, 5Ø hz
F8F8  JSR $F9B3   ..............
F8FB  JSR $EC@@   ..............    effacer mémoire (remplir de #4Ø)
..............    F935  STA BFDF
..............    F938  LDA #4Ø     premier caractère=#4Ø
..............    F93A  STA A@@@    et décaler #A@@@ --> #A@@1, à partir du bas
..............    F93D  LDX #17
..............    F93F  JSR $F982   c'est à dire remplir de #4Ø la mémoire
F8FE  LDA #ØØ     F942  LDA #ØØ
F9ØØ  STA Ø219    F944  STA Ø219    curseur HIRES en Ø,Ø
F9Ø3  STA Ø21A    F947  STA Ø21A
F9Ø6  STA 1Ø      F94A  STA 1Ø
F9Ø8  LDA #AØ     F94C  LDA #AØ
F9ØA  STA 11      F94E  STA 11      placer son adresse (#AØØØ)
F9ØC  LDA #2Ø     F95Ø  LDA #2Ø     et son motif
F9ØE  STA Ø215    F952  STA Ø215
F911  LDA #FF     F955  LDA #FF
F913  STA Ø213    F957  STA Ø213    PATTERN plein pour l'instant
F916  JSR $F8A9   F95A  JSR $F8DC   créer la fenêtre TEXT
F919  LDA #Ø1     F95D  LDA #Ø1     et mettre le curseur
F91B  ORA Ø26A    F95F  ORA Ø26A
F91E  STA Ø26A    F962  STA Ø26A
F921  PLA         F965  PLA
F922  RTS         F966  RTS
```

CREER L'ECRAN TEXT

```
F923  PHA          F967  PHA          sauver A
F924  LDA #FE      F968  LDA #FE
F926  AND Ø26A     F96A  AND Ø26A
F929  STA Ø26A     F96D  STA Ø26A     enlever le curseur
F92C  LDX #11      F970  LDX #11
F92E  JSR $F93E    F972  JSR $F982
décaler les caractères
F931  JSR $F97F    F975  JSR $F9C9    placer les paramètres de l'écran
F934  LDA #Ø1      F978  LDA #Ø1
F936  ORA Ø26A     F97A  ORA Ø26A
F939  STA Ø26A     F97D  STA Ø26A     et remettre le curseur
F93C  PLA          F980  PLA
F93D  RTS          F981  RTS
```

DECALER UNE ZONE D'APRES TABLE

```
F93E  LDY #Ø6      F982  LDY #Ø6      indiquer 6 paramètres
F940  LDA F94E,X   F984  LDA F992,X   prendre paramètres
F943  STA Ø1FF,Y   F987  STA ØØØB,Y   dans zone de travail
F946  DEX          F98A  DEX
F947  DEY          F98B  DEY
F948  BNE F940     F98C  BNE F984
F94A  JSR $EBD9    F98E  JSR $EDC4    faire le décalage
F94D  RTS          F991  RTS
```

Données caractères ROM --) RAM

```
F94E ....  BYT #FC7Ø,#B5ØØ,#Ø3ØØ
.... F992  BYT #FC78,#B5ØØ,#Ø3ØØ
```

Données caractères TEXT --) HIRES

```
F954 F99B  BYT #B4ØØ,#98ØØ,#Ø78Ø
```

Données HIRES --) TEXT

```
F95A F95E  BYT #98ØØ,#B4ØØ,#Ø78Ø
```

Données remplir écran HIRES

```
.... F9A4  BYT #AØØØ,#AØØ1,#1FBF
```

INITIALISER LE VIA 6522

```
F960  LDA #FF      F9AA  LDA #FF
F962  STA 0303     F9AC  STA 0303    port A (colonnes clavier) tout en sortie
F965  LDA #F7      F9AF  LDA #F7
F967  STA 0302     F9B1  STA 0302    port B en sortie,sauf b3 (LS....)
F96A  LDA #B7      F9B4  LDA #B7
F96C  STA 0300     F9B6  STA 0300    ouvrir le relais (b6=0)
F96F  LDA #DD      F9B9  LDA #DD     initialier PCR
F971  STA 030C     F9BB  STA 030C
F974  LDA #7F      F9BE  LDA #7F     pour l'instant,ne pas générer d'IRQ
F976  STA 030E     F9C0  STA 030E
F979  LDA #00      F9C3  LDA #00     T1 en mode monostable
F97B  STA 030B     F9C5  STA 030B
F97E  RTS          F9C8  RTS
```

CONFIGURER EN MODE TEXT

```
F97F  LDA #1A      F9C9  LDA #1A
F981  JSR $F9B3    F9CB  JSR $FA07   envoyer attribut TEXT 50 hz
F984  LDA #20      F9CE  LDA #20     effacer la ligne de statut
F986  LDY #28      F9D0  LDY #28
F988  STA BB7F,Y   F9D2  STA BB7F,Y
F98B  DEY          F9D5  DEY
F98C  BNE F988     F9D6  BNE F9D2
F98E  LDA #00      F9D8  LDA #00     indiquer mode TEXT
F990  STA 021F     F9DA  STA 021F
F993  STA 0269     ....  ....        placer le curseur en 0,1
F996  LDA #01      ....  ....        utilité à démontrer...
F998  STA 0268     ....  ....
F99B  LDA #80      ....  ....
F99D  STA 026D     ....  ....        adresse ligne 0=#BB80
F9A0  LDA #BB      ....  ....
F9A2  STA 026E     ....  ....
....  ....         F9DD  LDA #BB
....  ....         F9DF  STA 027B
....  ....         F9E2  STA 0279
....  ....         F9E5  LDA #A8
....  ....         F9E7  STA 027A    adresse ligne 1=#BBA8
....  ....         F9EA  LDA #D0
....  ....         F9EC  STA 0278    adresse ligne 2=#BBD0
F9A5  LDA #1B      F9EF  LDA #1B
F9A7  STA 026F     F9F1  STA 027E    l'écran a 27 lignes utiles...
```

```
.............. F9F4 LDA #04
.............. F9F6 STA 027D
.............. F9F9 LDA #10     et il y a #0410 caractères à scroller
.............. F9FB STA 027C    (26*40=1040=#410)
F9AA LDX #0C    F9FE LDX #0C    effacer l'écran
F9AC JSR $F409  FA00 JSR $0238
F9AF JSR $F729  FA03 JSR $F75A  et placer CAPS si nécessaire
F9B2 RTS        FA06 RTS
```

ENVOYER UN ATRIBUT DE MODE D'ECRAN

```
F9B3 STA BFDF   FA07 STA BFDF   envoyer l'attribut en bas de l'écran
F9B6 LDA #02    FA0A LDA #02    et placer timer 2
F9B8 LDX #00    FA0C LDX #00
F9BA LDY #03    FA0E LDY #03    pour attendre 3/100 ème de seconde
F9BC JSR $EBDC  FA10 JSR $EEC9
F9BF RTS        FA13 RTS
```

TESTER LA MEMOIRE

```
F9C0 JSR $F9DD  .............. tester 16 K/48 K
F9C3 JSR $FA06  .............. tester les 16 K ou 48 K
F9C6 BNE F9DC   .............. si erreur, on sort Z=0
F9C8 LDA #00    ..............
F9CA STA 0C     ..............
F9CC LDA #10    ..............
F9CE STA 0D     .............. placer index sur #1000 (??)
F9D0 LDA #00    .............. code=0
F9D2 JSR $FA49  .............. et tester si 0 accepté sur la page 10
F9D5 BNE F9DC   .............. si erreur, on sort, Z=0
F9D7 LDA #FF    .............. code=#FF
F9D9 JSR $FA49  .............. et tester si #FF accepté sur la page #10
F9DC RTS        ..............
```

Tester 16 K/48 K

Sortie: #2E1-#2E2 contient le haut de la mémoire (#3FFF/#BFFF)

```
F9DD LDA #00    ..............
F9DF STA 1000   .............. mettre 0 en #1000 (??)
F9E2 LDA #FF    ..............
F9E4 STA 5000   .............. et #FF en #5000
```

```
F9E7  LDA 1000  ..............  récupérer valeur en #1000
F9EA  BNE F9F6  ..............  si pas 0,c'est un 16 K
F9EC  LDA #00   ..............
F9EE  STA 0220  ..............  indiquer 48 K
F9F1  LDA #BF   ..............  si 48 K,top=#BFFF
F9F3  JMP $F9FD ..............
F9F6  LDA #01   ..............
F9F8  STA 0220  ..............  indiquer 16 K
F9FB  LDA #3F   ..............
F9FD  STA 02E2  ..............  et top=#3FFF
FA00  LDA #FF   ..............
FA02  STA 02E1  ..............  et sauver top poids faible aussi
FA05  RTS       ..............
```

Tester toute la mémoire

```
FA06  CLC       ..............
FA07  LDA 02E1  ..............
FA0A  ADC #01   ..............  ajuster valeur limite+1
FA0C  STA 0E    ..............
FA0E  LDA 02E2  ..............
FA11  ADC #00   ..............
FA13  STA 0F    ..............  dans #0E-#0F
FA15  LDA #00   ..............
FA17  STA 0C    ..............
FA19  LDA #04   ..............
FA1B  STA 0D    ..............  initialiser pointeur de départ à #0400
FA1D  LDY #00   ..............  et index
FA1F  LDA 0E    ..............
FA21  CMP 0C    ..............  tester si fin,poids faible
FA23  BNE FA2B  ..............
FA25  LDA 0F    ..............
FA27  CMP 0D    ..............  et éventuellement poids fort
FA29  BEQ FA48  ..............  sortir Z=1 si fin atteinte
FA2B  LDA #AA   ..............  motif %10101010
FA2D  STA (0C),Y ............  on écrit...
FA2F  LDA (0C),Y ............  on relit...
FA31  CMP #AA   ..............  c'est pareil ?
FA33  BNE FA48  ..............  non,sortir,Z=0
FA35  LDA #55   ..............  oui,on essaye avec motif %01010101
FA37  STA (0C),Y ............  on écrit
FA39  LDA (0C),Y ............  puis relit
FA3B  CMP #55   ..............  et enfin teste
FA3D  BNE FA48  ..............  erreur si pas pareil
```

```
FA3F  INC ØC      ..............  suivant poids faible
FA41  BNE FA45    ..............
FA43  INC ØD      ..............  et poids fort peut être...
FA45  JMP $FA1F   ..............  et on repart
FA48  RTS         ..............  peut être que #FAGØ est trop loin ?
```

Tester toute une page

```
FA49  STA ØE      ..............  sauver code de remplissage
FA4B  LDY #ØØ     ..............  initialiser index
FA4D  STA (ØC),Y  ..............  écrire le code
FA4F  DEY         ..............
FA5Ø  BNE FA4D    ..............  et finir toute la page
FA52  JSR $FA61   ..............  attendre environ 7,5 ms
FA55  LDY #ØØ     ..............  indexer la page à nouveau
FA57  LDA (ØC),Y  ..............  prendre ce qu'il reste
FA59  CMP ØE      ..............  et comparer à ce qu'on a mis
FA5B  BNE FA6Ø    ..............  si pas pareil, on sort Z=Ø
FA5D  DEY         ..............
FA5E  BNE FA57    ..............  si page OK, on sort Z=1
FA6Ø  RTS         ..............
```

Attendre 7684 microsecondes

```
FA61  LDX #Ø6     ..............
FA63  LDY #ØØ     ..............
FA65  DEY         ..............
FA66  BNE FA65    ..............
FA68  DEX         ..............
FA69  BNE FA65    ..............
FA6B  RTS         ..............
```

Remarque: curieuse manière de tester le mode 16/48K en #5ØØ ! Si l'adresse #5ØØ n'est pas en RAM, où seront stockés les programmes BASIC ?

```
..............  FA14  LDY #ØØ
..............  FA16  STY Ø26Ø     indiquer pas d'erreur de mémoire
..............  FA19  STY Ø22Ø     indiquer 48 K
..............  FA1C  STY Ø5ØØ     placer un Ø (!)
..............  FA1F  STY ØE
..............  FA21  DEY          Y=#FF
..............  FA22  STY ØC       poids faible pointeur
..............  FA24  STY 45ØØ     placer en #45ØØ aussi
..............  FA27  LDA Ø5ØØ     récupérer valeur laissée en #5ØØ
```

```
..............  FA2A  BNE FA30    sauter si pas Ø
..............  FA2C  LDA #CØ     48 K:top=#C000
..............  FA2E  BNE FA35    inconditionnel
..............  FA30  INC 0220    indiquer 16 K
..............  FA33  LDA #40     top=#4000
..............  FA35  STA ØF      et sauver dans limite
..............  FA37  INY         Y=#00
..............  FA38  LDA #03     debut =#3FF (on commence par incrémenter)
..............  FA3A  STA ØD
..............  FA3C  INC ØC      passer octet suivant
..............  FA3E  BNE FA42
..............  FA40  INC ØD
..............  FA42  LDA ØC
..............  FA44  CMP ØE      est-ce fini ?
..............  FA46  BNE FA4E
..............  FA48  LDA ØD
..............  FA4A  CMP ØF
..............  FA4C  BEQ FA5D    oui,on sort
..............  FA4E  LDA #AA     motif=%10101010
..............  FA50  STA (ØC),Y  on écrit
..............  FA52  CMP (ØC),Y  on lit pour comparer
..............  FA54  BNE FA5D    si pas bon,on sort
..............  FA56  LSR A       motif=%01010101
..............  FA57  STA (ØC),Y  on écrit
..............  FA59  CMP (ØC),Y  on lit pour comparer
..............  FA5B  BEQ FA3C    si c'est bon,on recommence
..............  FA5D  SEC
..............  FA5E  LDA ØF
..............  FA60  SBC #28     prendre top mémoire (16 K ou 48 K)
..............  FA62  STA ØF      et enlever #2800
..............  FA64  LDA ØE      pour calculer le haut en mode RELEASE
..............  FA66  CMP ØC
..............  FA68  LDA ØF
..............  FA6A  SBC ØD      et comparer à la valeur réellement atteinte
..............  FA6C  BCC FA77
..............  FA6E  LDA ØC      si celle-ci est en dessous,c'est elle
..............  FA70  LDY ØD      qui est le haut de la mémoire
..............  FA72  INC 0260    et on indique erreur de mémoire
..............  FA75  BNE FA7B    inconditionnel
..............  FA77  LDA ØE      prendre haut de mémoire
..............  FA79  LDY ØF      mode RELEASE
..............  FA7B  STA A6
..............  FA7D  STY A7      et sauver comme HIMEM
..............  FA7F  STA 02C1
```

```
.............. FA82  STY #2C2    et comme référence pour GRAB
.............. FA85  RTS
```

P) ROUTINES SONORES

1-Procédures standards

Bien entendu, il s'agit d'accéder facilement au PSG 8912.

Le PSG est organisé, pour le son, en 13 registres (voir leur signification dans le chapitre entrées/sorties). Tous ces registres sont latchés, c'est à dire qu'ils gardent leur valeur tant qu'on ne la remplace pas par une autre.

L'émission d'un son va donc consister à définir une période (sa hauteur), son canal, l'éventuel mixage du bruit, ainsi éventuellement qu'un choix d'enveloppe.

On aura tout intérêt à utiliser la routine qui initialise d'un coup les 14 registres du PSG (#FA6C/#FA86) avec les valeurs désirées. Un cas typique d'utilisation est celui du PING par exemple.

Ensuite, si on veut faire varier ce son, en volume par exemple, il suffira de modifier uniquement le registre voulume du canal considéré, ce qui est très rapide. Il convient pour celà d'utiliser la routine #F535/#F59Ø. Une illustration de ce procédé est la commande ZAP, qui fait varier la période du son émis.

Le principe est simple, et l'Oric n'attend plus que votre talent !

2-Listing

ENVOYER 14 PARAMETRES AU PSG 8912

Entrée: XY pointe sur l'adresse des 14 paramètres.

Sortie: P inchangé.

```
FA6C  PHP         FA86  PHP         sauver P
FA6D  SEI         FA87  SEI         inhiber IRQ
FA6E  STX 14      FA88  STX 14
FA7Ø  STY 15      FA8A  STY 15      sauver adresse pour indirection
FA72  LDY #ØØ     FA8C  LDY #ØØ     paramètre Ø
FA74  LDA (14),Y  FA8E  LDA (14),Y  prendre le paramètre
FA76  TAX         FA9Ø  TAX         dans X
FA77  TYA         FA91  TYA         numéro du registre
```

```
FA78  PHA          FA92  PHA          dans A et on sauve
FA79  JSR $F535    FA93  JSR $F590   envoyer le paramètre au GI
FA7C  PLA          FA96  PLA
FA7D  TAY          FA97  TAY          récupérer index
FA7E  INY          FA98  INY          et passer au suivant
FA7F  CPY #0E      FA99  CPY #0E      si les 14 transférés
FA81  BNE FA74     FA9B  BNE FA8E
FA83  PLP          FA9D  PLP          on sort.
FA84  RTS          FA9E  RTS
```

'PING' (COMMANDE)

```
FA85  LDX #8D      FA9F  LDX #A7
FA87  LDY #FA      FAA1  LDY #FA      indexer données pour PING
FA89  JSR $FA6C    FAA3  JSR $FA86    et envoyer au GI
FA8C  RTS          FAA6  RTS
```

Données pour PING

Fréquence=2800 Hz

```
FA8D FAA7  BYT #18,#00,#00,#00,#00,#00,#00
FA94 FAAE  BYT #3E,#10,#00,#00,#00,#0F,#00
```

'SHOOT' (COMMANDE)

```
FA9B  LDX #A3      FAB5  LDX #BD
FA9D  LDY #FA      FAB7  LDY #FA      indexer données pour SHOOT
FA9F  JSR $FA6C    FAB9  JSR $FA86    et envoyer
FAA2  RTS          FABC  RTS
```

Données pour SHOOT

```
FAA3 FABD  BYT #00,#00,#00,#00,#00,#00,#0F
FAAA FAC4  BYT #07,#10,#10,#10,#00,#08,#00
```

'EXPLODE' (COMMANDE)

```
FAB1  LDX #B9      FACB  LDX #D3
FAB3  LDY #FA      FACD  LDY #FA      indexer données pour EXPLODE
FAB5  JSR $FA6C    FACF  JSR $FA86    et envoyer
```

```
FAB8  RTS          FAD2  RTS
```

Données pour SHOOT

```
FAB9 FAD3  BYT #00,#00,#00,#00,#00,#00,#1F
FAC8 FADA  BYT #07,#10,#10,#10,#00,#18,#00
```

'ZAP' (COMMANDE)

Principe: pour obtenir l'effet voulu, on fait varier la fréquence du maxi (ultra-son, environ 65000 Hz) jusque vers 558 Hz (médium), en attendant 1,3 ms entre chaque fréquence. ZAP prend environ 150 ms pour s'exécuter.

```
FAC7  LDX #EC      FAE1  LDX #06
FAC9  LDY #FA      FAE3  LDY #FB      indexer données pour ZAP
FACB  JSR $FA6C    FAE5  JSR $FA86    et envoyer
FACE  LDA #00      FAE8  LDA #00      au début,fréquence maxi (+65000 Hz !)
FAD0  TAX          FAEA  TAX          pourquoi pas LDX #00 ?
FAD1  TXA          FAEB  TXA          on sauve la période sur la pile
FAD2  PHA          FAEC  PHA
FAD3  LDA #00      FAED  LDA #00      on indexe le premier registre (période)
FAD5  JSR $F535    FAEF  JSR $F590    et on envoie.
FAD8  LDX #00      FAF2  LDX #00
FADA  DEX          FAF4  DEX
FADB  BNE FADA     FAF5  BNE FAF4     attendre environ 1,3 ms
FADD  PLA          FAF7  PLA          récupérer période
FADE  TAX          FAF8  TAX          dans X
FADF  INX          FAF9  INX          et on diminue la fréquence
FAE0  CPX #70      FAFA  CPX #70      a-t-on atteint 558 Hz ?
FAE2  BNE FAD1     FAFC  BNE FAEB     non, on continue
FAE4  LDA #08      FAFE  LDA #08      indexer le volume du canal 1
FAE6  LDX #00      FB00  LDX #00      très silencieux...
FAE8  JSR $F535    FB02  JSR $F590    et éteindre le son
FAEB  RTS          FB05  RTS
```

Données pour ZAP

```
FAEC FB06  BYT #00,#00,#00,#00,#00,#00,#3E
FAF3 FB0D  BYT #0F,#00,#00,#00,#00,#00
```

SON CLAVIER NORMAL

```
FAFA  LDX #02     FB14  LDX #1C
FAFC  LDY #FB     FB16  LDY #FB     indexer données
FAFE  JSR $FA6C   FB18  JSR $FA86   et envoyer
FB01  RTS         FB1B  RTS

FB02 FB1C  BYT #1F,#30,#00,#00,#00,#00,#00
FB09 FB23  BYT #3E,#10,#00,#00,#1F,#00,#00
```

SON CLAVIER CONTROLE

```
FB10  LDX #18     FB2A  LDX #32
FB12  LDY #FB     FB2C  LDY #FB     indexer données
FB14  JSR $FA6C   FB2E  JSR $FA86   et envoyer
FB17  RTS         FB31  RTS

FB18 FB32  BYT #2F,#00,#00,#00,#00,#00,#00
FB1F FB39  BYT #3E,#10,#00,#00,#1F,#00,#00
```

'SOUND' (COMMANDE)

Entrée: #2E1-#2E2: numéro du canal
#2E3-#2E4: période demandée
#2E5-#2E6: volume

Sortie: #2E1 incrémenté si erreur dans les paramètres.

Programmation: la routine est écrite trois fois, une fois pour chaque canal ! Autant dire qu'elle devrait être trois fois plus courte, avec l'utilisation d'un index. Du travail baclé...

```
FB26  LDA 02E1    FB40  LDA 02E1
FB29  CMP #01     FB43  CMP #01     Canal 1 demandé ?
FB2B  BNE FB4F    FB45  BNE FB69    non,on saute
```

Traiter canal 1

```
FB2D  LDA #00     FB47  LDA #00     indexer période canal 1 (registre 0)
FB2F  LDX 02E3    FB49  LDX 02E3    prendre poids faible période
FB32  JSR $F535   FB4C  JSR $F590   et envoyer dans le GI 8912
FB35  LDA #01     FB4F  LDA #01     indexer période canal 1 (registre 1)
```

```
FB37  LDX Ø2E4    FB51  LDX Ø2E4    prendre poids fort
FB3A  JSR $F535   FB54  JSR $F59Ø   et envoyer
FB3D  LDA Ø2E5    FB57  LDA Ø2E5    prendre volume
FB4Ø  AND #ØF     FB5A  AND #ØF     ramener à Ø-15
FB42  BNE FB48    FB5C  BNE FB62
FB44  LDX #1Ø     FB5E  LDX #1Ø     si Ø,alors c'est 16 (silence) (LDA #1Ø ?)
FB46  BNE FB49    FB6Ø  BNE FB63    inconditionnel
FB48  TAX         FB62  TAX         volume dans X
FB49  LDA #Ø8     FB63  LDA #Ø8     indexer volume canal 1 (registre 8)
FB4B  JSR $F535   FB65  JSR $F59Ø   et envoyer
FB4E  RTS         FB68  RTS

FB4F  CMP #Ø2     FB69  CMP #Ø2     est-ce le canal 2 ?
FB51  BNE FB75    FB6B  BNE FB8F    non,sauter

Traiter canal 2

FB53  LDA #Ø2     FB6D  LDA #Ø2
FB55  LDX Ø2E3    FB6F  LDX Ø2E3
FB58  JSR $F535   FB72  JSR $F59Ø   mettre période poids faible
FB5B  LDA #Ø3     FB75  LDA #Ø3
FB5D  LDX Ø2E4    FB77  LDX Ø2E4
FB6Ø  JSR $F535   FB7A  JSR $F59Ø   puis poids fort
FB63  LDA Ø2E5    FB7D  LDA Ø2E5
FB66  AND #ØF     FB8Ø  AND #ØF
FB68  BNE FB6E    FB82  BNE FB88
FB6A  LDX #1Ø     FB84  LDX #1Ø
FB6C  BNE FB6F    FB86  BNE FB89
FB6E  TAX         FB88  TAX
FB6F  LDA #Ø9     FB89  LDA #Ø9
FB71  JSR $F535   FB8B  JSR $F59Ø   et enfin volume
FB74  RTS         FB8E  RTS

FB75  CMP #Ø3     FB8F  CMP #Ø3     est-ce la canal 3 ?
FB77  BNE FB9B    FB91  BNE FBB5    non,sauter

Traiter le canal 3

FB79  LDA #Ø4     FB93  LDA #Ø4
FB7B  LDX Ø2E3    FB95  LDX Ø2E3
FB7E  JSR $F535   FB98  JSR $F59Ø   envoyer période poids faible
FB81  LDA #Ø5     FB9B  LDA #Ø5
FB83  LDX Ø2E4    FB9D  LDX Ø2E4
FB86  JSR $F535   FBAØ  JSR $F59Ø   période poids fort
```

```
FB89  LDA Ø2E5    FBA3  LDA Ø2E5
FB8C  AND #ØF     FBA6  AND #ØF
FB8E  BNE FB94    FBA8  BNE FBAE
FB9Ø  LDX #1Ø     FBAA  LDX #1Ø
FB92  BNE FB95    FBAC  BNE FBAF
FB94  TAX         FBAE  TAX
FB95  LDA #ØA     FBAF  LDA #ØA
FB97  JSR $F535   FBB1  JSR $F59Ø   et enfin volume
FB9A  RTS         FBB4  RTS
```

Traiter canaux bruit

```
FB9B  LDA #Ø6     FBB5  LDA #Ø6     indexer période du bruit
FB9D  LDX Ø2E3    FBB7  LDX Ø2E3    et prendre sa valeur
FBAØ  JSR $F535   FBBA  JSR $F59Ø   et envoyer au GI 8912
FBA3  LDA Ø2E1    FBBD  LDA Ø2E1    reprendre canal
FBA6  CMP #Ø4     FBCØ  CMP #Ø4     c'est le 4 ?
FBA8  BEQ FB3D    FBC2  BEQ FB57    oui,placer volume du canal 1
FBAA  CMP #Ø5     FBC4  CMP #Ø5     c'est le 5 ?
FBAC  BEQ FB63    FBC6  BEQ FB7D    oui,placer le volume du canal 2
FBAE  CMP #Ø6     FBC8  CMP #Ø6     c'est le 6 ?
FBBØ  BEQ FB89    FBCA  BEQ FBA3    oui,placer le volume du canal 3
FBB2  INC Ø2EØ    FBCC  INC Ø2EØ    non,erreur
FBB5  RTS         FBCF  RTS
```

'PLAY' (COMMANDE)

Entrée: #2E1-#2E2:canaux de bruit
#2E3-#2E4:canaux de son
#2E5-#2E6:numéro de l'enveloppe
#2E7-#2E8:période de l'enveloppe

Sortie: #2EØ incrémenté si erreur de paramètre

```
FBB6  LDA Ø2E3    FBDØ  LDA Ø2E3    prendre canal précisé dans bØ-b1-b2
FBB9  ASL A       FBD3  ASL A
FBBA  ASL A       FBD4  ASL A
FBBB  ASL A       FBD5  ASL A       et tranférer dans b3-b4-b5
FBBC  ORA Ø2E1    FBD6  ORA Ø2E1    ajouter en bØ-b1-b2 les canaux bruits
FBBF  EOR #3F     FBD9  EOR #3F     inverser car actif à Ø
FBC1  TAX         FBDB  TAX         dans X
FBC2  LDA #Ø7     FBDC  LDA #Ø7     indexer le registre d'autorisation
FBC4  JSR $F535   FBDE  JSR $F59Ø   et envoyer au GI 8912
```

```
FBC7  CLC          FBE1  CLC
FBC8  LDA Ø2E7     FBE2  LDA Ø2E7     calculer la période
FBCB  ASL A        FBE5  ASL A        (ASL #Ø2E7/ROL #Ø2E8 était trop simple ?)
FBCC  STA Ø2E7     FBE6  STA Ø2E7
FBCF  LDA Ø2E8     FBE9  LDA Ø2E8
FBD2  ROL A        FBEC  ROL A
FBD3  STA Ø2E8     FBED  STA Ø2E8
FBD6  LDA #ØB      FBFØ  LDA #ØB      indexer période d'enveloppe (regitre 11)
FBD8  LDX Ø2E7     FBF2  LDX Ø2E7     poids faible
FBDB  JSR $F535    FBF5  JSR $F59Ø    et envoyer
FBDE  LDA #ØC      FBF8  LDA #ØC
FBEØ  LDX Ø2E8     FBFA  LDX Ø2E8
FBE3  JSR $F535    FBFD  JSR $F59Ø    idem poids fort enveloppe
FBE6  LDA Ø2E5     FCØØ  LDA Ø2E5     prendre No d'enveloppe
FBE9  AND #Ø7      FCØ3  AND #Ø7      et ramener à Ø-7
FBEB  TAY          FCØ5  TAY          comme index
FBEC  LDA FBF6,Y   FCØ6  LDA FC1Ø,Y   prendre numéro réel pour le GI 8912
FBEF  TAX          FCØ9  TAX          LDX adresse,Y sera inventé plus tard !
FBFØ  LDA #ØD      FCØA  LDA #ØD      indexer registre d'enveloppe (registre 13)
FBF2  JSR $F535    FCØC  JSR $F59Ø    et envoyer au 8912
FBF5  RTS          FCØF  RTS
```

Table d'équivalence des enveloppes. Au passage on perd l'enveloppe No11, qui aurait été plus utile que de répéter la Ø !

```
FBF6 FC1Ø  BYT #ØØ,#ØØ,#Ø4,#Ø8
FBFA FC14  BYT #ØA,#ØB,#ØC,#ØD
```

'MUSIC' (COMMANDE)

Entrée: #2E1-#2E2: canal
#2E3-#2E4: octave
#2E5-#2E6: note
#2E7-#2E8: volume

Sortie: #2EØ incrémenté si erreur

```
FBFE  LDX #E1      FC18  LDX #E1      indexer le canal
FCØØ  LDA #Ø4      FC1A  LDA #Ø4
FCØ2  JSR $EC09    FC1C  JSR $F2E4    vérifier canal pas plus grand que 3
FCØ5  BCS FC4Ø     FC1F  BCS FC5A     si oui,erreur
FCØ7  LDX #E3      FC21  LDX #E3      indexer l'octave
FCØ9  LDA #Ø8      FC23  LDA #Ø8
```

```
FC0B  JSR $EC06   FC25  JSR $F2F8   et vérifier pas plus grand que 7
FC0E  BCS FC40    FC28  BCS FC5A    si oui,erreur
FC10  LDX #E5     FC2A  LDX #E5     indexer note
FC12  LDA #0D     FC2C  LDA #0D
FC14  JSR $EC09   FC2E  JSR $F2E4   et vérifier pas plus grande que 12
FC17  BCS FC40    FC31  BCS FC5A    oui,erreur
FC19  LDY 02E3    FC33  LDY 02E3    prendre octave
FC1C  LDX 02E5    FC36  LDX 02E5    prendre nombre comme index
FC1F  LDA FC44,X  FC39  LDA FC5E,X  prendre poids fort période
FC22  STA 02E4    FC3C  STA 02E4    et sauver
FC25  LDA FC51,X  FC3F  LDA FC6B,X  poids faible période
FC28  STA 02E3    FC42  STA 02E3    on sauve aussi
FC2B  LDA 02E7    FC45  LDA 02E7    prendre volume
FC2E  STA 02E5    FC48  STA 02E5    à la même place que pour SOUND
FC31  DEY         FC4B  DEY         compter les octaves
FC32  BMI FC3D    FC4C  BMI FC57    c'est le bon,on sort
FC34  LSR 02E4    FC4E  LSR 02E4    sinon,on divise la période par 2
FC37  ROR 02E3    FC51  ROR 02E3    poids faible aussi
FC3A  JMP $FC31   FC54  JMP $FC4B   et on recommence
FC3D  JMP $FB26   FC57  JMP $FB40   mettre le volume comme SOUND
FC40  INC 02E0    FC5A  INC 02E0    indiquer erreur
FC43  RTS         FC5D  RTS

Table des périodes des notes.Poids fort

FC44 FC5E  BYT #03,#07,#07,#06,#06,#05
FC4B FC55  BYT #05,#05,#04,#04,#04,#04,#03

Table des périodes des notes.Poids faible

FC51 FC6B  BYT #00,#77,#0B,#A6,#47,#EC
FC58 FC71  BYT #97,#47,#FB,#B3,#70,#30,#F4
```

En fait, on fourni au PSG non pas la période mais la période divisée par 16. Le calcul de la fréquence se fait donc par: F=1 E+6/P/16, et donc la période: P=1 E+6/F/16

Les fréquences sont données ci après pour quelques octaves seulement.

Note:	DO	DO #	RE	RE #	MI	FA	FA #	SOL	SOL #	LA	LA #	SI
Numéro:	1	2	3	4	5	6	7	8	9	10	11	12
Lettre:	C		D		E	F		G		A		B

```
Freq 0: 32   34   36   38   41   43   46   49   51   55   58   61
Freq 3: 261  277  293  311  329  349  370  392  414  440  466  494
Freq 4: 523  554  587  622  659  698  740  784  831  880  932  988
Freq 7: 4186 4437 4700 4978 5277 5590 5921 6274 6650 7042 7462 7904
```

N'IMPORTE QUOI DIVERS

```
FC5E  JSR $F40F   ................
FC61  TXA         ................
FC62  BPL $FC67   ................
FC64  STX 02DF    ................
FC67  JMP $ED4F   ................
FC6A ....  BYT #4C,#67,#B3,#4C,#7B,#B3
```

FC70 FC78 dessin des caractères (CF annexe)

TABLEAU DE CONVERSION CLAVIER/ASCII

Structure: voir routine #F494/#F4EF

```
FF70 FF78  BYT '7'    ,'j'+$80,'m'+$80,'k'+$80,' '    ,'u'+$80,'y'+$80,'8'
FF78 FF80  BYT 'n'+$80,'t'+$80,'6'    ,'9'    ,','    ,'i'+$80,'h'+$80,'l'+$80
FF80 FF88  BYT '5'    ,'r'+$80,'b'+$80,';'    ,'.'    ,'o'+$80,'g'+$80,'0'
FF88 FF90  BYT 'v'+$80,'f'+$80,'4'    ,'-'    ,' VT'  ,'p'+$80,'e'+$80,'/'
FF90 FF98  BYT 'NUL'  ,'NUL'  ,'NUL'  ,'NUL'  ,'NUL'  ,'NUL'  ,'NUL'  ,'NUL'
FF98 FFA0  BYT '1'    ,'ESC'  ,'z'+$80,'NUL'  ,' BS'  ,'DEL'  ,'a'+$80,' CR'
FFA0 FFA8  BYT 'x'+$80,'q'+$80,'2'    ,'\'    ,' LF'  ,']'    ,'s'+$80,'NUL'
FFA8 FFB0  BYT '3'    ,'d'+$80,'c'+$80,"'"    ,' HT'  ,'['    ,'w'+$80,'='

FFB0 FFB8  BYT '&'    ,'J'    ,'M'    ,'K'    ,' '    ,'U'    ,'Y'    ,'*'
FFB8 FFC0  BYT 'N'    ,'T'    ,'^'    ,'('    ,'<'    ,'I'    ,'H'    ,'L'
FFC0 FFC8  BYT '%'    ,'R'    ,'B'    ,':'    ,')'    ,'O'    ,'G'    ,'}'
FFC8 FFD0  BYT 'V'    ,'F'    ,'$'    ,'_'    ,' VT'  ,'P'    ,'E'    ,'?'
FFD0 FFD8  BYT 'NUL'  ,'NUL'  ,'NUL'  ,'NUL'  ,'NUL'  ,'NUL'  ,'NUL'  ,'NUL'
FFD8 FFE0  BYT '!'    ,'ESC'  ,'Z'    ,'NUL'  ,' BS'  ,'DEL'  ,'A'    ,' CR'
FFE0 FFE8  BYT 'X'    ,'Q'    ,'@'    ,'"'    ,' LF'  ,'}'    ,'S'    ,'NUL'
FFE8 FFF0  BYT '#'    ,'D'    ,'C'    ,'?'    ,' HT'  ,'{'    ,'W'    ,'+'

FFF0 ....  BYT #A0,#E3,#B7,#BA,#A4,#A3,#A0
FFF8 ....  BYT #B0,#C4
.... FFF8  BYT #D0,#81
```

VECTEURS D'INTERRUPTIONS

```
FFFA .... BYT #0228   NMI
.... FFFA BYT #0247   NMI
FFFC .... BYT #F42D   RESET
.... FFFC BYT #F88F   RESET
FFFE .... BYT #0228   IRQ
.... FFFE BYT #0244   IRQ
```

Temps de prise en compte d'une interruption:

Il est des applications où le temps de prise en compte d'une interruption peut créer des problèmes insurmontables. Ces caractéristiques sont hélas assez peu précisées dans les ouvrages, il est bon de faire le point ici.

Le texte fait référence à l'IRQ, la seule accessible facilement, mais l'ensemble peut s'appliquer au cas des NMI.

Le 6502 reconnait un niveau bas sur la ligne IRQ, à ne pas confondre avec le VIA qui est capable de générer des IRQ sur des fronts particuliers de ses lignes de contrôle.

Le 6502 attendant la fin d'une instruction pour tester l'IRQ (de la même manière que le BASIC ne teste le Ctrl C qu'à la fin d'une instruction), il y aura un retard de 0 à 7 microsecondes entre le moment où la ligne IRQ passe à l'état bas et le moment où le 6502 la prend en compte. Il convient donc de même de laisser la ligne à l'état bas pendant plus de 7 microsecondes, sinon le 6502 riquerait de rater une IRQ.

Il faut ensuite 6 microsecondes pour sauver sur la pile PCL,PCH et P (il faut trois fois sauver la valeur et incrémenter S, ce qui donne 3*(1+1)

Il faut ensuite aller chercher la valeur du vecteur, ce qui nécessite 2 micros secondes (1 pour le poids faible, et 1 pour le poids fort)

Et enfin, effectuer le JMP en RAM, ce qui nécessite 3 microsecondes supplémentaires.

Si l'on ajoute qu'il faut finir par RTI, qui fait 6 microsecondes, c'est donc (0 à 7)+6+2+3+6=17 à 24 microsecondes obligatoirement perdues à chaque IRQ ou NMI ! Difficile dans ces conditions de travailler très rapidement, en vidéo par exemple. Quoique...

Il faut souvent prendre en compte le délai supplémentaire introduit par le VIA par exemple: entre une transition sur CA1 et la génération de l'IRQ,il peut y avoir 0 à 2 microsecondes (à moins de se synchroniser sur l'horloge). Mais ce délais supplémentaire est différent car il ne rallonge pas le temps de traitement réel, mais le diffère seulement.

On peut souvent jouer sur la caractéristiques de certains boitiers (VIA 6522, FDC 1793, FDC 1771 etc...) de garder la ligne à l'état bas tant que l'interruption n'est pas traitée. Ceci permet de rattraper sans dommage une IRQ plus longue que les autres (par exemple, toutes les 256 IRQ's, l'incrémentation

du poids fort d'un pointeur pourra dépasser légèrement la longueur maxi, dépassement ratrappé par les IRQ's suivantes un peu plus courtes.)

Bref, si vous devez avoir une réponse très rapide (et fréquente), l'IRQ n'est pas forcément la solution.

Un exemple typique est celui de l'interfaçage d'un lecteur de disquette, qui envoie (ou reçoit) un octet toute les 32 microsecondes, ce qui laisse avec les IRQ seulement 8 à 15 microsecondes (11 à 18 en modifiant directement le vecteur) pour traiter l'information, ce qui est (très) critique. Un choix plus judicieux est la lecture d'un bit sur un port d'E/S, bit facile à tester (b7 ou b6). Il suffit donc d'une boucle LDA/BPL par exemple, qui nécessite seulement 6 ou 7 microsecondes, c'est à dire dans le cas présent 25 ou 26 microsecondes pour traiter la donnée, ce qui est plus confortable.

Signalons enfin une méthode rarement utilisable, mais qui permet de gagner 4 microsecondes: mettre un CLI dans les IRQ, de sorte qu'une nouvelle IRQ viendra interrompre l'IRQ courante. On gagne globalement 4 microsecondes (6 du RTI moins 2 du CLI) et une quasi certitude de saturer la pile, ce qui pose des problèmes sérieux. C'est la solution de la dernière chance, mais il vaut mieux reconsidérer la question au niveau Hard, si c'est possible !

En résumé, avant de concevoir une interface qui enverra beaucoup de données, il est préférable d'écrire les routines de lecture, et surtout d'estimer précisément leur temps de réponse. Il semblerait qu'une IRQ toute les 32 microsecondes soit assez proche de la limite pour ce pauvre 6502 à 1 Mhz.

CODE:20-		CODE:21-!		CODE:22-"		CODE:23-#		CODE:24-$	
V1.0	00	V1.0 X	08	V1.0 X X	14	V1.0 X X	14	V1.0 X	08
FC70	00	FC78 X	08	FC80 X X	14	FC88 X X	14	FC90 XXXX	1E
V1.1	00	V1.1 X	08	V1.1 X X	14	V1.1 XXXXX	3E	V1.1 X X	28
FC78	00	FC80 X	08	FC88	00	FC90 X X	14	FC98 XXX	1C
text	00	text X	08	text	00	text XXXXX	3E	text X X	0A
B500	00	B508	00	B510	00	B518 X X	14	B520 XXXX	3C
hirs	00	hirs X	08	hirs	00	hirs X X	14	hirs X	08
9900	00	9908	00	9910	00	9918	00	9920	00

CODE:25-%		CODE:26-&		CODE:27-'		CODE:28-(		CODE:29-)	
V1.0 XX	30	V1.0 X	10	V1.0 X	08	V1.0 X	08	V1.0 X	08
FC98 XX X	32	FCA0 X X	28	FCA8 X	08	FCB0 X	10	FCB8 X	04
V1.1 X	04	V1.1 X X	28	V1.1 X	08	V1.1 X	20	V1.1 X	02
FCA0 X	08	FCA8 X	10	FCB0	00	FCB8 X	20	FCC0 X	02
text X	10	text X X X	2A	text	00	text X	20	text X	02
B528 X XX	26	B530 X X	24	B538	00	B540 X	10	B548 X	04
hirs XX	06	hirs XX X	1A	hirs	00	hirs X	08	hirs X	08
9928	00	9930	00	9938	00	9940	00	9948	00

CODE:2A-*

```
V1.Ø   X    Ø8
FCCØ X X X  2A
V1.1  XXX   1C
FCC8   X    Ø8
text  XXX   1C
B55Ø X X X  2A
hirs   X    Ø8
995Ø        Ø3
```

CODE:2B-+

```
V1.Ø        ØØ
FCC8   X    Ø8
V1.1   X    Ø8
FCDØ XXXXX  3E
text   X    Ø8
B558   X    Ø8
hirs        ØØ
9958        ØØ
```

CODE:2C-,

```
V1.Ø        ØØ
FCDØ        ØØ
V1.1        ØØ
FCD8        ØØ
text        ØØ
B56Ø   X    Ø8
hirs   X    Ø8
9960 X      1Ø
```

CODE:2D--

```
V1.Ø        ØØ
FCD8        ØØ
V1.1        ØØ
FCEØ XXXXX  3E
text        ØØ
B568        ØØ
hirs        ØØ
9968        ØØ
```

CODE:2E-.

```
V1.Ø        ØØ
FCEØ        ØØ
V1.1        ØØ
FCE8        ØØ
text        ØØ
B57Ø     X  Ø4
hirs        ØØ
997Ø        ØØ
```

CODE:2F-/

```
V1.Ø        ØØ
FCE8     X  Ø2
V1.1   X    Ø4
FCFØ   X    Ø8
text  X     1Ø
B578 X      2Ø
hirs        ØØ
9978        ØØ
```

CODE:3Ø-Ø

```
V1.Ø  XXX   1C
FCFØ X   X  22
V1.1 X  XX  26
FCF8 X X X  2A
text XX  X  32
B58Ø X   X  22
hirs  XXX   1C
9980        ØØ
```

CODE:31-1

```
V1.Ø   X    Ø8
FCF8  XX    18
V1.1   X    Ø8
FDØØ   X    Ø8
text   X    Ø8
B588   X    Ø8
hirs  XXX   1C
9988        ØØ
```

CODE:32-2

```
V1.Ø  XXX   1C
FDØØ X   X  22
V1.1     X  Ø2
FDØ8     X  Ø4
text     X  Ø8
B59Ø   X    1Ø
hirs XXXXX  3E
9990        ØØ
```

CODE:33-3

```
V1.Ø XXXXX  3E
FDØ8     X  Ø2
V1.1     X  Ø4
FD1Ø    XX  ØC
text     X  Ø2
B598 X   X  22
hirs  XXX   1C
9998        ØØ
```

CODE:34-4

```
V1.Ø    X   Ø4
FD1Ø   XX   ØC
V1.1  X X   14
FD18 X   X  24
text XXXXX  3E
B5AØ    X   Ø4
hirs    X   Ø4
99AØ        ØØ
```

CODE:35-5

```
V1.Ø XXXXX  3E
FD18 X      2Ø
V1.1 XXXX   3C
FD2Ø     X  Ø2
text     X  Ø2
B5A8 X   X  22
hirs  XXX   1C
99A8        ØØ
```

CODE:36-6

```
V1.Ø   XX   ØC
FD2Ø   X    1Ø
V1.1 X      2Ø
FD28 XXXX   3C
text   X    Ø8
B5BØ X   X  22
hirs  XXX   1C
99BØ        ØØ
```

CODE:37-7

```
V1.Ø XXXXX  3E
FD28     X  Ø2
V1.1     X  Ø4
FD3Ø     X  Ø8
text     X  1Ø
B5B8     X  1Ø
hirs     X  1Ø
99B8        ØØ
```

CODE:38-8

```
V1.Ø  XXX   1C
FD3Ø X   X  22
V1.1 X   X  22
FD38  XXX   1C
text X   X  22
B5CØ X   X  22
hirs  XXX   1C
99CØ        ØØ
```

CODE:39-9

```
V1.Ø  XXX   1C
FD38 X   X  22
V1.1 X   X  22
FD4Ø  XXXX  1E
text     X  Ø2
B5C8     X  Ø4
hirs  XX    18
99C8        ØØ
```

CODE:3A-:

```
V1.Ø        ØØ
FD4Ø        ØØ
V1.1   X    Ø8
FD48        ØØ
text        ØØ
B5DØ   X    Ø8
hirs        ØØ
99DØ        ØØ
```

CODE:3B-;

```
V1.Ø        ØØ
FD48        ØØ
V1.1   X    Ø8
FD5Ø        ØØ
text        ØØ
B5D8   X    Ø8
hirs   X    Ø8
99D8 X      1Ø
```

CODE:3C-<

```
V1.Ø    X   Ø4
FD5Ø   X    Ø8
V1.1   X    1Ø
FD58 X      2Ø
text   X    1Ø
B5E8    X   Ø8
hirs     X  Ø4
99E8        ØØ
```

CODE:3D-=

```
V1.Ø        ØØ
FD58        ØØ
V1.1 XXXXX  3E
FD6Ø        ØØ
text XXXXX  3E
B5E8        ØØ
hirs        ØØ
99E8        ØØ
```

```
CODE:3E-)       CODE:3F-?       CODE:40-à       CODE:41-A       CODE:42-B

V1.Ø  X     1Ø  V1.Ø  XXX   1C  V1.Ø  XXX   1C  V1.Ø   X    Ø8  V1.Ø XXXX   3C
FD6Ø   X    Ø8  FD68 X   X 22  FD7Ø X   X 22  FD78  X X  14  FD8Ø X   X 22
V1.1    X   Ø4  V1.1    X   Ø4  V1.1 X X X 2A  V1.1 X   X 22  V1.1 X   X 22
FD68     X  Ø2  FD7Ø   X     Ø8  FD78 X XXX 2E  FD8Ø X   X 22  FD88 XXXX   3C
text    X   Ø4  text   X     Ø8  text X XX  2C  text XXXXX 3E  text X   X 22
B5FØ   X    Ø8  B5F8          ØØ  B6ØØ X     2Ø  B6Ø8 X   X 22  B61Ø X   X 22
hirs  X     1Ø  hirs   X     Ø8  hirs  XXXX  1E  hirs X   X 22  hirs XXXX   3C
99FØ            ØØ  99F8          ØØ  9AØØ          ØØ  9AØ8          ØØ  9A1Ø          ØØ


CODE:43-C       CODE:44-D       CODE:45-E       CODE:46-F       CODE:47-G

V1.Ø  XXX   1C  V1.Ø XXXX   3C  V1.Ø XXXXX 3E  V1.Ø XXXXX 3E  V1.Ø  XXXX  1E
FD88 X   X 22  FD9Ø X   X 22  FD98 X     2Ø  FDAØ X     2Ø  FDA8 X     2Ø
V1.1 X     2Ø  V1.1 X   X 22  V1.1 X     2Ø  V1.1 X     2Ø  V1.1 X     2Ø
FD9Ø X     2Ø  FD98 X   X 22  FDAØ XXXX  3C  FDA8 XXXX  3C  FDBØ X     2Ø
text X     2Ø  text X   X 22  text X     2Ø  text X     2Ø  text X  XX 26
B618 X   X 22  B62Ø X   X 22  B628 X     2Ø  B63Ø X     2Ø  B638 X   X 22
hirs  XXX   1C  hirs XXXX   3C  hirs XXXXX 3E  hirs X     2Ø  hirs  XXXX  1E
9A18            ØØ  9A2Ø          ØØ  9A28          ØØ  9A3Ø          ØØ  9A38          ØØ


CODE:48-H       CODE:49-I       CODE:4A-J       CODE:4B-K       CODE:4C-L

V1.Ø X   X 22  V1.Ø  XXX   1C  V1.Ø      X Ø2  V1.Ø X   X 22  V1.Ø X     2Ø
FDBØ X   X 22  FDB8    X     Ø8  FDCØ      X Ø2  FDC8 X  X  24  FDDØ X     2Ø
V1.1 X   X 22  V1.1    X     Ø8  V1.1      X Ø2  V1.1 X X    28  V1.1 X     2Ø
FDB8 XXXXX 3E  FDCØ    X     Ø8  FDC8      X Ø2  FDDØ XX     3Ø  FDD8 X     2Ø
text X   X 22  text    X     Ø8  text      X Ø2  text X X    28  text X     2Ø
B64Ø X   X 22  B648    X     Ø8  B65Ø X   X 22  B658 X  X  24  B66Ø X     2Ø
hirs X   X 22  hirs  XXX   1C  hirs  XXX   1C  hirs X   X 22  hirs XXXXX 3E
9A4Ø            ØØ  9A48          ØØ  9A5Ø          ØØ  9A58          ØØ  9A6Ø          ØØ


CODE:4D-M       CODE:4E-N       CODE:4F-O       CODE:5Ø-P       CODE:51-Q

V1.Ø X   X 22  V1.Ø X   X 22  V1.Ø  XXX   1C  V1.Ø XXXX  3C  V1.Ø  XXX   1C
FDD8 XX XX 36  FDEØ X   X 22  FDE8 X   X 22  FDFØ X   X 22  FDF8 X   X 22
V1.1 X X X 2A  V1.1 XX  X 32  V1.1 X   X 22  V1.1 X   X 22  V1.1 X   X 22
FDEØ X X X 2A  FDE8 X X X 2A  FDFØ X   X 22  FDF8 XXXX  3C  FE0Ø X   X 22
text X   X 22  text X  XX 26  text X   X 22  text X     2Ø  text X X X 2A
B668 X   X 22  B67Ø X   X 22  B678 X   X 22  B68Ø X     2Ø  B688 X  X  24
hirs X   X 22  hirs X   X 22  hirs  XXX   1C  hirs X     2Ø  hirs  XX X 1A
9A68            ØØ  9A7Ø          ØØ  9A78          ØØ  9A8Ø          ØØ  9A88          ØØ
```

CODE:52-R

```
V1.Ø XXXX   3C
FEØØ X   X  22
V1.1 X   X  22
FEØ8 XXXX   3C
text X X    2Ø
B69Ø X   X  24
hirs X   X  22
9A9Ø        ØØ
```

CODE:53-S

```
V1.Ø  XXX   1C
FEØ8 X   X  22
V1.1 X      2Ø
FE1Ø  XXX   1C
text     X  Ø2
B698 X   X  22
hirs  XXX   1C
9A98        ØØ
```

CODE:54-T

```
V1.Ø XXXXX  3E
FE1Ø   X    Ø8
V1.1   X    Ø8
FE18   X    Ø8
text   X    Ø8
B6AØ   X    Ø8
hirs   X    Ø8
9AAØ        ØØ
```

CODE:55-U

```
V1.Ø X   X  22
FE18 X   X  22
V1.1 X   X  22
FE2Ø X   X  22
text X   X  22
B6A8 X   X  22
hirs  XXX   1C
9AA8        ØØ
```

CODE:56-V

```
V1.Ø X   X  22
FE2Ø X   X  22
V1.1 X   X  22
FE28 X   X  22
text X   X  22
B6BØ  X X   14
hirs   X    Ø8
9ABØ        ØØ
```

CODE:57-W

```
V1.Ø X   X  22
FE28 X   X  22
V1.1 X   X  22
FE3Ø X X X  2A
text X X X  2A
B6B8 XX XX  36
hirs X   X  22
9AB8        ØØ
```

CODE:58-X

```
V1.Ø X   X  22
FE3Ø X   X  22
V1.1  X X   14
FE38   X    Ø8
text  X X   14
B6C0 X   X  22
hirs   X    Ø8
9AC8        ØØ
```

CODE:59-Y

```
V1.Ø X   X  22
FE38 X   X  22
V1.1   X    Ø4
FE4Ø   X    Ø8
text   X    Ø8
B6C8   X    Ø8
hirs   X    Ø8
9AD8        ØØ
```

CODE:5A-Z

```
V1.Ø XXXXX  3E
FE4Ø     X  Ø2
FE48   X    1Ø
FE5Ø   X    1Ø
text   X    1Ø
B6DØ X      2Ø
hirs XXXXX  3E
9ADØ        ØØ
```

CODE:5B-'

```
V1.Ø  XXXX  1E
FE48   X    1Ø
V1.1   X    1Ø
FE5Ø   X    1Ø
text   X    1Ø
B6D8   X    1Ø
hirs  XXXX  1E
9AD8        ØØ
```

CODE:5C-£

```
V1.Ø        ØØ
FE5Ø X      2Ø
V1.1  X     1Ø
FE58   X    Ø8
text     X  Ø4
B6EØ      X Ø2
hirs        ØØ
9AEØ        ØØ
```

CODE:5D-§

```
V1.Ø XXXX   3C
FE58     X  Ø4
V1.1     X  Ø4
FE6Ø     X  Ø4
text     X  Ø4
B6E8     X  Ø4
hirs XXXX   3C
9AE8        ØØ
```

CODE:5E-^

```
V1.Ø   X    Ø8
FE6Ø  X X   14
V1.1 X X X  2A
FE68   X    Ø8
text   X    Ø8
B6FØ   X    Ø8
hirs   X    Ø8
9AFØ        ØØ
```

CODE:5F-_

```
V1.Ø   XXX  ØE
FE68   X    1Ø
V1.1   X    1Ø
FE7Ø   X    1Ø
text XXXX   3C
B6F8   X    1Ø
hirs XXXXX  3E
9AF8        ØØ
```

CODE:60-'

```
V1.Ø   XX   ØC
FE7Ø  X  X  12
V1.1 X XX X 2D
FE78 X X  X 29
text X X  X 29
B7ØØ X XX X 2D
hirs  X  X  12
9B ØØ  XX   ØC
```

CODE:61-a

```
V1.Ø        ØØ
FE78        ØØ
V1.1  XXX   1C
FE8Ø     X  Ø2
text  XXXX  1E
B7Ø8 X   X  22
hirs  XXXX  1E
9B Ø8       ØØ
```

CODE:62-b

```
V1.Ø X      2Ø
FE8Ø X      2Ø
V1.1 XXXX   3C
FE88 X   X  22
text X   X  22
B71Ø X   X  22
hirs XXXX   3C
9B1Ø        ØØ
```

CODE:63-c

```
V1.Ø        ØØ
FE88        ØØ
V1.1  XXXX  1E
FE9Ø X      2Ø
text X      2Ø
B718 X      2Ø
hirs  XXXX  1E
9B18        ØØ
```

CODE:64-d

```
V1.Ø     X  Ø2
FE9Ø     X  Ø2
V1.1  XXXX  1E
FE98 X   X  22
text X   X  22
B72Ø X   X  22
hirs  XXXX   E
9B2Ø        ØØ
```

CODE:65-e

```
V1.Ø        ØØ
FE98        ØØ
V1.1  XXX   1C
FEAØ X   X  22
text XXXXX  3E
B728 X      2Ø
hirs  XXXX  1E
9B28        ØØ
```

CODE:66-f

```
V1.Ø   XX   ØC
FEAØ  X  X  12
V1.1  X     1Ø
FEA8 XXXX   3C
text  X     1Ø
B73Ø  X     1Ø
hirs  X     1Ø
9B3Ø        ØØ
```

CODE:67-g

```
V1.Ø        ØØ
FEA8        ØØ
V1.1  XXX   1C
FEBØ X   X  22
text X   X  22
B738  XXXX  1E
hirs     X  Ø2
9B38  XXX   1C
```

COI .:68-h

```
V1.Ø X      2Ø
FEBØ X      2Ø
V1.1 XXXX   3C
FEB8 X   X  22
text X   X  22
B748 X   X  22
hirs X   Y  22
9B4Ø        ØØ
```

CODE:69-i

```
V1.Ø   X    Ø8
FEB8        ØØ
V1.1  XX    18
FEC8  X     Ø8
text  X     Ø8
B758  X     Ø8
hirs XXX    1C
9B48        ØØ
```

CODE:6A-j

```
V1.Ø    X   Ø4
FECØ        ØØ
V1.1   XX   ØC
FEC8    X   Ø4
text    X   Ø4
B75Ø    X   Ø4
hirs X  X   24
9B5Ø  XX    18
```

CODE:6B-k

```
V1.Ø Y      2Ø
FEC8 X      2Ø
V1.1 X   X  22
FED8  X     Ø8
text XXX    38
B768 X   X  24
hirs X   X  22
9B58        ØØ
```

CODE:6C-l

```
V1.Ø  XX    18
FED8   X    Ø8
V1.1   X    Ø8
FEDØ X  X   24
text   X    Ø8
B76Ø   X    Ø8
hirs XXX    1C
9B6Ø        ØØ
```

CODE:6D-m

```
V1.Ø        ØØ
FED8        ØØ
V1.1 XX XX  36
FEEØ X X X  2A
text X X X  2A
B768 X X X  2A
hirs X   X  22
9B68        ØØ
```

CODE:6E-n

```
V1.Ø        ØØ
FEE8        ØØ
V1.1 XXXX   3C
FEE8 X   X  22
text X   X  22
B778 X   X  22
hirs X   X  22
9B7Ø        ØØ
```

CODE:6F-o

```
V1.Ø        ØØ
FEE8        ØØ
V1.1  XXX   1C
FEFØ X   X  22
text X   X  22
B778 X   X  22
hirs  XXX   1C
9B78        ØØ
```

CODE:7Ø-p

```
V1.Ø        ØØ
FEF8        ØØ
V1.1 XXXX   3C
FEF8 X   X  22
text X   X  22
B78Ø XXXX   3C
hirs X      2Ø
9B8Ø X      2Ø
```

CODE:71-q

```
V1.Ø        ØØ
FEF8        ØØ
V1.1  XXXX  1E
FFØØ X   X  22
text X   X  22
B788  XXXX  1E
hirs     X  Ø2
9B88     X  Ø2
```

CODE:72-r

```
V1.Ø        ØØ
FFØØ        ØØ
V1.1 X XXX  2E
FF8 XX      3Ø
text X      2Ø
B79Ø X      2Ø
hirs X      2Ø
9B9Ø        ØØ
```

CODE:73-s

```
V1.Ø        ØØ
FF8         ØØ
V1.1  XXXX  1E
FF1Ø X      2Ø
text  XXX   1C
B798     X  Ø2
hirs XXXX   3C
9B98        ØØ
```

CODE:74-t

```
V1.Ø  X     1Ø
FF1Ø  X     1Ø
V1.1 XXXX   3C
FF18  X     1Ø
text  X     1Ø
B7AØ  X  X  12
hirs   XX   ØC
9BAØ        ØØ
```

CODE:75-u

```
V1.Ø        ØØ
FF18        ØØ
V1.1 X   X  22
FF2Ø X   X  22
text X   X  22
B7A8 X  XX  26
hirs  XX X  1A
9BA8        ØØ
```

CODE:76-v

```
V1.Ø        ØØ
FF2Ø        ØØ
V1.1 X   X  22
FF28 X   X  22
text X   X  22
B7BØ  X X   14
hirs   X    Ø8
9BBØ        ØØ
```

CODE:77-w

```
V1.Ø        ØØ
FF28        ØØ
V1.1 X   X  22
FF3Ø X   X  22
text X X X  2A
B7B8 X X X  2A
hirs XX XX  36
9BB8        ØØ
```

CODE:78-x

```
V1.Ø        ØØ
FF3Ø        ØØ
V1.1 X   X  22
FF38  X X   14
text   X    Ø8
B7CØ  X X   14
hirs X   X  22
9BCØ        ØØ
```

CODE:79-y

```
V1.Ø        ØØ
FF38        ØØ
V1.1 X   X  22
FF4Ø X   X  22
text X   X  22
B7C8  XXXX  1E
hirs     X  Ø2
9BC8  XXX   1C
```

CODE:7A-z CODE:7B-é CODE:7C-ù CODE:7D-è CODE:7E-ê

```
V1.Ø      ØØ V1.Ø  XXX ØE V1.Ø  X   Ø8 V1.Ø XXX   38 V1.Ø X X X  2A
FF4Ø      ØØ FF48  (X  18 FF5Ø  X   Ø8 FF58  XX  ØC FF6Ø  X X X 15
V1.1 XXXXX 3E V1.1 XX   18 V1.1 X   Ø8 V1.1 XX  ØC V1.1 X X X  2A
FF48   X  Ø4 FF5Ø XX    3Ø FF58 X   Ø8 FF6Ø  XX  Ø6 FF68  X X X 15
text   X  Ø8 text XX    18 text X   Ø8 text XX  ØC text X X X  2A
B7DØ   X  1Ø B7D8 XX    18 B7EØ X   Ø8 B7E8  XX  ØC B7FØ  X X X 15
hirs XXXXX 3E hirs  XXX ØE hirs X   Ø8 hirs XXX  38 hirs X X X  2A
9BDØ      ØØ 9BD8       ØØ 9BEØ X   Ø8 9BE8      ØØ 9BFØ  X X X 15
```

CODE:7F

```
V1.Ø XXXXXX
FF68 XXXXXX
V1.1 XXXXXX
FF7Ø XXXXXX
text XXXXXX
B7F8 XXXXXX
hirs XXXXXX
9BF3 XXXXXX
```

CODE:2Ø- CODE:21-! CODE:22-" CODE:23-# CODE:24-$

```
         ØØ      XXX   38       XXX Ø7     XXXXXX 3F          ØØ
         ØØ      XXX   38       XXX Ø7     XXXXXX 3F          ØØ
         ØØ      XXX   38       XXX Ø7     XXXXXX 3F          ØØ
         ØØ            ØØ           ØØ           ØØ     XXX  38
text     ØØ text       ØØ text      ØØ text      ØØ text XXX  38
B9ØØ     ØØ B9Ø8       ØØ B91Ø      ØØ B918      ØØ B92Ø      ØØ
hirs     ØØ hirs       ØØ hirs      ØØ hirs      ØØ hirs      ØØ
9DØØ     ØØ 9DØ8       ØØ 9D1Ø      ØØ 9D18      ØØ 9D2Ø      ØØ
```

CODE:25-% CODE:26-& CODE:27-' CODE:28-(CODE:29-)

```
     XXX   38         XXX Ø7     XXXXXX 3F          ØØ     XXX   38
     XXX   38         XXX Ø7     XXXXXX 3F          ØØ     XXX   38
     XXX   38         XXX Ø7     XXXXXX 3F          ØØ     XXX   38
     XXX   38     XXX   38     XXX   38       XXX Ø7        XXX Ø7
text XXX   38 text XXX   38 text XXX   38 text   XXX Ø7 text   XXX Ø7
B928       ØØ B93Ø       ØØ B938      ØØ B94Ø      ØØ B948      ØØ
hirs       ØØ hirs       ØØ hirs      ØØ hirs      ØØ hirs      ØØ
9D28       ØØ 9D3Ø       ØØ 9D38      ØØ 9D4Ø      ØØ 9D48      ØØ
```

CODE:2A-*

```
      XXX 07
      XXX 07
      XXX 07
      XXX 07
text  XXX 07
B950      00
hirs      00
9D50      00
```

CODE:2B-+

```
      XXXXXX 3F
      XXXXXX 3F
      XXXXXX 3F
         XXX 07
text     XXX 07
B958      00
hirs      00
9D58      00
```

CODE:2C-,

```
               00
               00
               00
      XXXXXX 3F
text XXXXXX 3F
B960      00
hirs      00
9D60      00
```

CODE:2D--

```
         XXX   38
         XXX   38
         XXX   38
      XXXXXX 3F
text XXXXXX 3F
B968      00
hirs      00
9D68      00
```

CODE:2E-.

```
         XXX 07
         XXX 07
         XXX 07
      XXXXXX 3F
text XXXXXX 3F
B970      00
hirs      00
9D70      00
```

CODE:2F-/

```
      XXXXXX 3F
      XXXXXX 3F
      XXXXXX 3F
      XXXXXX 3F
text XXXXXX 3F
B978      00
hirs      00
9D78      00
```

CODE:30-0

```
               00
               00
               00
               00
text           00
B980 XXX   38
hirs XXX   38
9D80 XXX   38
```

CODE:31-1

```
         XXX   38
         XXX   38
         XXX   38
               00
text           00
B988 XXX   38
hirs XXX   38
9D88 XXX   38
```

CODE:32-2

```
         XXX 07
         XXX 07
         XXX 07
               00
text           00
B990 XXX   38
hirs XXX   38
9D90 XXX   38
```

CODE:33-3

```
      XXXXXX 3F
      XXXXXX 3F
      XXXXXX 3F
               00
text           00
B998 XXX   38
hirs XXX   38
9D98 XXX   38
```

CODE:34-4

```
               00
               00
               00
      XXX   38
text  XXX   38
B9A0 XXX   38
hirs XXX   38
9DA0 XXX   38
```

CODE:35-5

```
         XXX   38
         XXX   38
         XXX   38
      XXX   38
text  XXX   38
B9A8 XXX   38
hirs XXX   38
9DA8 XXX   38
```

CODE:36-6

```
            XXX 07
            XXX 07
            XXX 07
         XXX   38
text     XXX   38
B9B0 XXX   38
hirs XXX   38
9DB0 XXX   38
```

CODE:37-7

```
         XXXXXX 3F
         XXXXXX 3F
         XXXXXX 3F
      XXX   38
text  XXX   38
B9B8 XXX   38
hirs XXX   38
9DB8 XXX   38
```

CODE:38-8

```
               00
               00
               00
            XXX 07
text        XXX 07
B9C0 XXX   38
hirs XXX   38
9DC0 XXX   38
```

CODE:39-9

```
      XXX   38
      XXX   38
      XXX   38
         XXX 07
text     XXX 07
B9C8 XXX   38
hirs XXX   38
9DC8 XXX   38
```

CODE:3A-:

```
            XXX 07
            XXX 07
            XXX 07
            XXX 07
text        XXX 07
B9D0 XXX   38
hirs XXX   38
9DD0 XXX   38
```

CODE:3B-;

```
         XXXXXX 3F
         XXXXXX 3F
         XXXXXX 3F
            XXX 07
text        XXX 07
B9D8 XXX   38
hirs XXX   38
9DD8 XXX   38
```

CODE:3C-<

```
               00
               00
               00
      XXXXXX 3F
text XXXXXX 3F
B9E0 XXX   38
hirs XXX   38
9DE0 XXX   38
```

CODE:3D-=

```
         XXX   38
         XXX   38
         XXX   38
      XXXXXX 3F
text XXXXXX 3F
B9E8 XXX   38
hirs XXX   38
9DE8 XXX   38
```

CODE:3E-)	CODE:3F-?	CODE:40-à	CODE:41-A	CODE:42-B
XXX 07	XXXXXX 3F	00	XXX 38	XXX 07
XXX 07	XXXXXX 3F	00	XXX 38	XXX 07
XXX 07	XXXXXX 3F	00	XXX 38	XXX 07
XXXXXX 3F	XXXXXX 3F	00	00	00
text XXXXXX 3F	text XXXXXX 3F	text 00	text 00	text 00
B9F5 XXX 38	B9FD XXX 38	BA06 XXX 07	BA0E XXX 07	BA16 XXX 07
hirs XXX 38	hirs XXX 38	hirs XXX 07	hirs XXX 07	hirs XXX 07
9DF7 XXX 38	9DFF XXX 38	9E08 XXX 07	9E10 XXX 07	9E18 XXX 07

CODE:43-C	CODE:44-D	CODE:45-E	CODE:46-F	CODE:47-G
XXXXXX 3F	00	XXX 38	XXX 07	XXXXXX 3F
XXXXXX 3F	00	XXX 38	XXX 07	XXXXXX 3F
XXXXXX 3F	00	XXX 38	XXX 07	XXXXXX 3F
00	XXX 38	XXX 38	XXX 38	XXX 38
text 00	text XXX 38	text XXX 38	text XXX 38	text XXX 38
BA20 XXX 07	BA28 XXX 07	BA30 XXX 07	BA38 XXX 07	BA40 XXX 07
hirs XXX 07	hirs XXX 07	hirs XXX 07	hirs XXX 07	hirs XXX 07
9E20 XXX 07	9E28 XXX 07	9E30 XXX 07	9E38 XXX 07	9E40 XXX 07

CODE:48-H	CODE:49-I	CODE:4A-J	CODE:4B-K	CODE:4C-L
00	XXX 38	XXX 07	XXXXXX 3F	00
00	XXX 38	XXX 07	XXXXXX 3F	00
00	XXX 38	XXX 07	XXXXXX 3F	00
XXX 07	XXX 07	XXX 07	XXX 07	XXXXXX 3F
text XXX 07	text XXX 07	text XXX 07	text XXX 07	text XXXXXX 3F
BA48 XXX 07	BA50 XXX 07	BA58 XXX 07	BA60 XXX 07	BA68 XXX 07
hirs XXX 07	hirs XXX 07	hirs XXX 07	hirs XXX 07	hirs XXX 07
9E48 XXX 07	9E50 XXX 07	9E58 XXX 07	9E60 XXX 07	9E68 XXX 07

CODE:4D-M	CODE:4E-N	CODE:4F-O	CODE:50-P	CODE:51-Q
XXX 38	XXX 07	XXXXXX 3F	00	XXX 38
XXX 38	XXX 07	XXXXXX 3F	00	XXX 38
XXX 38	XXX 07	XXXXXX 3F	00	XXX 38
XXXXXX 3F	XXXXXX 3F	XXXXXX 3F	00	00
text XXXXXX 3F	text XXXXXX 3F	text XXXXXX 3F	text 00	text 00
BA70 XXX 07	BA78 XXX 07	BA80 XXX 07	BA88 XXXXXX 3F	BA90 XXXXXX 3F
hirs XXX 07	hirs XXX 07	hirs XXX 07	hirs XXXXXX 3F	hirs XXXXXX 3F
9E70 XXX 07	9E78 XXX 07	9E80 XXX 07	9E88 XXXXXX 3F	9E90 XXXXXX 3F

```
CODE:52-R       CODE:53-S       CODE:54-T       CODE:55-U       CODE:56-V

      XXX Ø7          XXXXXX 3F               ØØ      XXX    38       XXX Ø7
      XXX Ø7          XXXXXX 3F               ØØ      XXX    38       XXX Ø7
      XXX Ø7          XXXXXX 3F               ØØ      XXX    38       XXX Ø7
         ØØ              ØØ          XXX     38      XXX    38          XXX     38
text     ØØ  text        ØØ  text XXX     38  text XXX    38  text XXX     38
BA98 XXXXXX 3F  BAAØ XXXXXX 3F  BAA8 XXXXXX 3F  BABØ XXXXXX 3F  BAB8 XXXXXX 3F
hirs XXXXXX 3F  hirs XXXXXX 3F  hirs XXXXXX 3F  hirs XXXXXX 3F  hirs XXXXXX 3F
9E98 XXXXXX 3F  9EAØ XXXXXX 3F  9EA8 XXXXXX 3F  9EBØ XXXXXX 3F  9EB8 XXXXXX 3F


CODE:57-W       CODE:58-X       CODE:59-Y       CODE:5A-Z       CODE:5B-'

    XXXXXX 3F               ØØ      XXX    38       XXX Ø7      XXXXXX 3F
    XXXXXX 3F               ØØ      XXX    38       XXX Ø7      XXXXXX 3F
    XXXXXX 3F               ØØ      XXX    38       XXX Ø7      XXXXXX 3F
    XXX    38          XXX Ø7          XXX Ø7       XXX Ø7         XXX Ø7
text XXX    38  text    XXX Ø7  text    XXX Ø7  text    XXX Ø7  text    XXX Ø7
BAC0 XXXXXX 3F  BAC8 XXXXXX 3F  BAD0 XXXXXX 3F  BAD8 XXXXXX 3F  BAE0 XXXXXX 3F
hirs XXXXXX 3F  hirs XXXXXX 3F  hirs XXXXXX 3F  hirs XXXXXX 3F  hirs XXXXXX 3F
9EC0 XXXXXX 3F  9EC8 XXXXXX 3F  9ED0 XXXXXX 3F  9ED8 XXXXXX 3F  9EE0 XXXXXX 3F


CODE:5C-£       CODE:5D-§       CODE:5E-^       CODE:5F-_       CODE:60-'

         ØØ          XXX    38       XXX Ø7      XXXXXX 3F       X X X 55
         ØØ          XXX    38       XXX Ø7      XXXXXX 3F       X X X 55
         ØØ          XXX    38       XXX Ø7      XXXXXX 3F       X X X 55
    XXXXXX 3F       XXXXXX 3F      XXXXXX 3F       XXXXXX 3F       X X X 55
text XXXXXX 3F  text XXXXXX 3F  text XXXXXX 3F  text XXXXXX 3F  text  X X X 55
BAE0 XXXXXX 3F  BAF5 XXXXXX 3F  BAF0 XXXXXX 3F  BB05 XXXXXX 3F  BB0E  X X X 55
hirs XXXXXX 3F  hirs XXXXXX 3F  hirs XXXXXX 3F  hirs XXXXXX 3F  hirs  X X X 55
9EEF XXXXXX 3F  9EF7 XXXXXX 3F  9EFF XXXXXX 3F  9F07 XXXXXX 3F  9F10  X X X 55


CODE:61-a       CODE:62-b       CODE:63-c       CODE:64-d       CODE:65-e

      X X X 55        X X X 55        X X X 55        X X X 55        X X X 55
      X X X 55        X X X 55        X X X 55        X X X 55        X X X 55
      X X X 55        X X X 55        X X X 55        X X X 55        X X X 55
      X X X 55        X X X 55        X X X 55        X X X 55        X X X 55
text  X X X 55  text  X X X 55  text  X X X 55  text  X X X 55  text  X X X 55
BB18  X X X 55  BB20  X X X 55  BB28  X X X 55  BB30  X X X 55  BB38  X X X 55
hirs  X X X 55  hirs  X X X 55  hirs  X X X 55  hirs  X X X 55  hirs  X X X 55
9F18  X X X 55  9F20  X X X 55  9F28  X X X 55  9F30  X X X 55  9F38  X X X 55
```

```
CODE:66-f       CODE:67-g       CODE:68-h       CODE:69-i       CODE:6A-j

     X X X 55        X X X 55        X X X 55        X X X 55        X X X 55
     X X X 55        X X X 55        X X X 55        X X X 55        X X X 55
     X X X 55        X X X 55        X X X 55        X X X 55        X X X 55
     X X X 55        X X X 55        X X X 55        X X X 55        X X X 55
text X X X 55   text X X X 55   text X X X 55   text X X X 55   text X X X 55
BB40 X X X 55   BB48 X X X 55   BB50 X X X 55   BB58 X X X 55   BB60 X X X 55
hirs X X X 55   hirs X X X 55   hirs X X X 55   hirs X X X 55   hirs X X X 55
9F40 X X X 55   9F48 X X X 55   9F50 X X X 55   9F58 X X X 55   9F60 X X X 55
```

```
CODE:6B-k       CODE:6C-l       CODE:6D-m
     X X X 55        X X X 55        X X X 55
     X X X 55        X X X 55        X X X 55
     X X X 55        X X X 55        X X X 55
     X X X 55        X X X 55        X X X 55
text X X X 55   text X X X 55   text X X X 55
BB68 X X X 55   BB70 X X X 55   BB78 X X X 55
hirs X X X 55   hirs X X X 55   hirs X X X 55
9F68 X X X 55   9F70 X X X 55   9F78 X X X 55
```

CHAPITRE VII

ANNEXES

A) TABLES ET CONVERSIONS

1-Codes de controle et attributs vidéo

Abréviations:

C: clignotant
NC: non clignotant
JØ: caractères normaux
J1: caractères alternés
SH: simple hauteur
DH: double hauteur

ØØ	Ø	NUL		ØØ	8Ø	@	Encre noire
Ø1	1	SOH	A-a Recopie	Ø1	81	A-a	Encre rouge
Ø2	2	STX	B-b	Ø2	82	B-b	Encre verte
Ø3	3	ETX	C-c BREAK	Ø3	83	C-c	Encre jaune
Ø4	4	EOT	D-d Dbl hauteur	Ø4	84	D-d	Encre bleue
Ø5	5	ENQ	E-e	Ø5	85	E-e	Encre mauve
Ø6	6	ACK	F-f Son clavier	Ø6	86	F-f	Encre magenta
Ø7	7	BEL	G-g PING	Ø7	87	G-g	Encre blanche
Ø8	8	BS	H-h Curseur gauche	Ø8	88	H-h	JØ,NC,SH
Ø9	9	HT	I-i Curseur droite	Ø9	89	I-i	J1,NC,SH
ØA	1Ø	LF	J-j Curseur bas	ØA	8A	J-j	JØ,NC,DH
ØB	11	VT	K-k Curseur haut	ØB	8B	K-k	J1,NC,DH
ØC	12	FF	L-l Efface l'écran	ØC	8C	L-l	JØ,C,SH
ØD	13	CR	M-m Return/RC	ØD	8D	M-m	J1,C,SH
ØE	14	SO	N-n Efface ligne	ØE	8E	N-n	JØ,C,DH
ØF	15	SI	O-o Inhibe clavier	ØF	8F	O-o	J1,C,DH
1Ø	16	DLE	P-p Inverse b2 #26A	1Ø	9Ø	P-p	Papier noir
11	17	DC1	Q-q Curseur	11	91	Q-q	Papier rouge

```
12  18 DC2 R-r                      12  92 R-r   Papier vert
13  19 DC3 S-s Inhibe écran         13  93 S-s   Papier jaune
14  20 DC4 T-t Majuscule/min.       14  94 T-t   Papier bleu
15  21 NAK U-u                      15  95 U-u   Papier magenta
16  22 SYN V-v                      16  96 V-v   Papier Cyan
17  23 ETB W-w                      17  97 W-w   Papier blanc
18  24 CAN X-x Annule entrée        18  98 X-x   Text 60 Hz
19  25 EM  Y-y                      19  99 Y-y   Text 60 Hz
1A  26 SUB Z-z Attributs            1A  9A Z-z   Text 50 Hz
1B  27 ESC [-( Attributs            1B  9B [-(   Text 50 Hz
1C  28 FS  \                        1C  9C \-!   Hires 60 Hz
1D  29 GS  ]   Protect. col 0/1     1D  9D ]-)   Hires 60 Hz
1E  30 RS      Curseur en 0,1       1E  9E ^     Hires 50 Hz
1F  31 US  DEL                      1F  9F _-DEL Hires 50 Hz
```

2-Conversions numériques

Les colonnes representent dans l'ordre:
Hexadecimal,Decimal,Decimal*256,Binaire,Signe,Exposant et code 6502

```
00   0     0 00000000 +  0 +127 BRK
01   1   256 00000001 +  1 -128 ORA (LL,X)
02   2   512 00000010 +  2 -127
03   3   768 00000011 +  3 -126
04   4  1024 00000100 +  4 -125
05   5  1280 00000101 +  5 -124 ORA LL
06   6  1536 00000110 +  6 -123 ASL LL
07   7  1792 00000111 +  7 -122
08   8  2048 00001000 +  8 -121 PHP
09   9  2304 00001001 +  9 -120 ORA #DD
0A  10  2560 00001010 + 10 -119 ASL A
0B  11  2816 00001011 + 11 -118
0C  12  3072 00001100 + 12 -117
0D  13  3328 00001101 + 13 -116 ORA HHLL
0E  14  3584 00001110 + 14 -115 ASL HHLL
0F  15  3840 00001111 + 15 -114
10  16  4096 00010000 + 16 -113 BPL DEP
11  17  4352 00010001 + 17 -112 ORA (LL),Y
12  18  4608 00010010 + 18 -111
13  19  4864 00010011 + 19 -110
14  20  5120 00010100 + 20 -109
15  21  5376 00010101 + 21 -108 ORA LL,X
```

```
16  22  5632 00010110 + 22 -107 ASL LL,X
17  23  5888 00010111 + 23 -106
18  24  6144 00011000 + 24 -105 CLC
19  25  6400 00011001 + 25 -104 ORA HHLL,Y
1A  26  6656 00011010 + 26 -103
1B  27  6912 00011011 + 27 -102
1C  28  7168 00011100 + 28 -101
1D  29  7424 00011101 + 29 -100 ORA HHLL,X
1E  30  7680 00011110 + 30 - 99 ASL HHLL,X
1F  31  7936 00011111 + 31 - 98
20  32  8192 00100000 + 32 - 97 JSR HHLL
21  33  8448 00100001 + 33 - 96 AND (LL,X)
22  34  8704 00100010 + 34 - 95
23  35  8960 00100011 + 35 - 94
24  36  9216 00100100 + 36 - 93 BIT LL
25  37  9472 00100101 + 37 - 92 AND LL
26  38  9728 00100110 + 38 - 91 ROL LL
27  39  9984 00100111 + 39 - 90
28  40 10240 00101000 + 40 - 89 PLP
29  41 10496 00101001 + 41 - 88 AND #DD
2A  42 10752 00101010 + 42 - 87 ROL A
2B  43 11008 00101011 + 43 - 86
2C  44 11264 00101100 + 44 - 85 BIT HHLL
2D  45 11520 00101101 + 45 - 84 AND HHLL
2E  46 11776 00101110 + 46 - 83 ROL HHLL
2F  47 12032 00101111 + 47 - 82
30  48 12288 00110000 + 48 - 81 BMI DEP
31  49 12544 00110001 + 49 - 80 AND (LL),Y
32  50 12800 00110010 + 50 - 79
33  51 13056 00110011 + 51 - 78
34  52 13312 00110100 + 52 - 77
35  53 13568 00110101 + 53 - 76 AND LL,X
36  54 13824 00110110 + 54 - 75 ROL LL,X
37  55 14080 00110111 + 55 - 74
38  56 14336 00111000 + 56 - 73 SEC
39  57 14592 00111001 + 57 - 72 AND HHLL,Y
3A  58 14848 00111010 + 58 - 71
3B  59 15104 00111011 + 59 - 70
3C  60 15360 00111100 + 60 - 69
3D  61 15616 00111101 + 61 - 68 AND HHLL,X
3E  62 15872 00111110 + 62 - 67 ROL HHLL,X
3F  63 16128 00111111 + 63 - 66
40  64 16384 01000000 + 64 - 65 RTI
41  65 16640 01000001 + 65 - 64 EOR (LL,X)
```

```
42  66 16896 01000010 + 66 - 63
43  67 17152 01000011 + 67 - 62
44  68 17408 01000100 + 68 - 61
45  69 17664 01000101 + 69 - 60 EOR LL
46  70 17920 01000110 + 70 - 59 LSR LL
47  71 18176 01000111 + 71 - 58
48  72 18432 01001000 + 72 - 57 PHA
49  73 18688 01001001 + 73 - 56 EOR #DD
4A  74 18944 01001010 + 74 - 55 LSR A
4B  75 19200 01001011 + 75 - 54
4C  76 19456 01001100 + 76 - 53 JMP HHLL
4D  77 19712 01001101 + 77 - 52 EOR HHLL
4E  78 19968 01001110 + 78 - 51 LSR HHLL
4F  79 20224 01001111 + 79 - 50
50  80 20480 01010000 + 80 - 49 BVC DEP
51  81 20736 01010001 + 81 - 48 EOR (LL),Y
52  82 20992 01010010 + 82 - 47
53  83 21248 01010011 + 83 - 46
54  84 21504 01010100 + 84 - 45
55  85 21760 01010101 + 85 - 44 EOR LL,X
56  86 22016 01010110 + 86 - 43 LSR LL,X
57  87 22272 01010111 + 87 - 42
58  88 22528 01011000 + 88 - 41 CLI
59  89 22784 01011001 + 89 - 40 EOR HHLL,Y
5A  90 23040 01011010 + 90 - 39
5B  91 23296 01011011 + 91 - 38
5C  92 23552 01011100 + 92 - 37
5D  93 23808 01011101 + 93 - 36 EOR HHLL,X
5E  94 24064 01011110 + 94 - 35 LSR HHLL,X
5F  95 24320 01011111 + 95 - 34
60  96 24576 01100000 + 96 - 33 RTS
61  97 24832 01100001 + 97 - 32 ADC (LL,X)
62  98 25088 01100010 + 98 - 31
63  99 25344 01100011 + 99 - 30
64 100 25600 01100100 +100 - 29
65 101 25856 01100101 +101 - 28 ADC LL
66 102 26112 01100110 +102 - 27 ROR LL
67 103 26368 01100111 +103 - 26
68 104 26624 01101000 +104 - 25 PLA
69 105 26880 01101001 +105 - 24 ADC #DD
6A 106 27136 01101010 +106 - 23 ROR A
6B 107 27392 01101011 +107 - 22
6C 108 27648 01101100 +108 - 21 JMP (HHLL)
6D 109 27904 01101101 +109 - 20 ADC HHLL
```

```
6E 110 28160 01101110 +110 - 19 ROR HHLL
6F 111 28416 01101111 +111 - 18
70 112 28672 01110000 +112 - 17 BVS DEP
71 113 28928 01110001 +113 - 16 ADC (LL),Y
72 114 29184 01110010 +114 - 15
73 115 29440 01110011 +115 - 14
74 116 29696 01110100 +116 - 13
75 117 29952 01110101 +117 - 12 ADC LL,X
76 118 30208 01110110 +118 - 11 ROR LL,X
77 119 30464 01110111 +119 - 10
78 120 30720 01111000 +120 -  9 SEI
79 121 30976 01111001 +121 -  8 ADC HHLL,Y
7A 122 31232 01111010 +122 -  7
7B 123 31488 01111011 +123 -  6
7C 124 31744 01111100 +124 -  5
7D 125 32000 01111101 +125 -  4 ADC HHLL,X
7E 126 32256 01111110 +126 -  3 ROR HHLL,X
7F 127 32512 01111111 +127 -  2
80 128 32768 10000000 -128 -  1
81 129 33024 10000001 -127 +  0 STA (LL,X)
82 130 33280 10000010 -126 +  1
83 131 33536 10000011 -125 +  2
84 132 33792 10000100 -124 +  3 STY LL
85 133 34048 10000101 -123 +  4 STA LL
86 134 34304 10000110 -122 +  5 STX LL
87 135 34560 10000111 -121 +  6
88 136 34816 10001000 -120 +  7 DEY
89 137 35072 10001001 -119 +  8
8A 138 35328 10001010 -118 +  9 TXA
8B 139 35584 10001011 -117 + 10
8C 140 35840 10001100 -116 + 11 STY HHLL
8D 141 36096 10001101 -115 + 12 STA HHLL
8E 142 36352 10001110 -114 + 13 STX HHLL
8F 143 36608 10001111 -113 + 14
90 144 36864 10010000 -112 + 15 BCC DEP
91 145 37120 10010001 -111 + 16 STA (LL),Y
92 146 37376 10010010 -110 + 17
93 147 37632 10010011 -109 + 18
94 148 37888 10010100 -108 + 19 STY LL,X
95 149 38144 10010101 -107 + 20 STA LL,X
96 150 38400 10010110 -106 + 21 STX LL,Y
97 151 38656 10010111 -105 + 22
98 152 38912 10011000 -104 + 23 TYA
99 153 39168 10011001 -103 + 24 STA HHLL,Y
```

```
9A 154 39424 10011010 -102 + 25 TXS
9B 155 39680 10011011 -101 + 26
9C 156 39936 10011100 -100 + 27
9D 157 40192 10011101 - 99 + 28 STA HHLL,X
9E 158 40448 10011110 - 98 + 29
9F 159 40704 10011111 - 97 + 30
A0 160 40960 10100000 - 96 + 31 LDY #DD
A1 161 41216 10100001 - 95 + 32 LDA (LL,X)
A2 162 41472 10100010 - 94 + 33 LDX #DD
A3 163 41728 10100011 - 93 + 34
A4 164 41984 10100100 - 92 + 35 LDY LL
A5 165 42240 10100101 - 91 + 36 LDA LL
A6 166 42496 10100110 - 90 + 37 LDX LL
A7 167 42752 10100111 - 89 + 38
A8 168 43008 10101000 - 88 + 39 TAY
A9 169 43264 10101001 - 87 + 40 LDA #DD
AA 170 43520 10101010 - 86 + 41 TAX
AB 171 43776 10101011 - 85 + 42
AC 172 44032 10101100 - 84 + 43 LDY HHLL
AD 173 44288 10101101 - 83 + 44 LDA HHLL
AE 174 44544 10101110 - 82 + 45 LDX HHLL
AF 175 44800 10101111 - 81 + 46
B0 176 45056 10110000 - 80 + 47 BCS DEP
B1 177 45312 10110001 - 79 + 48 LDA (LL),Y
B2 178 45568 10110010 - 78 + 49
B3 179 45824 10110011 - 77 + 50
B4 180 46080 10110100 - 76 + 51 LDY LL,X
B5 181 46336 10110101 - 75 + 52 LDA LL,X
B6 182 46592 10110110 - 74 + 53 LDX LL,Y
B7 183 46848 10110111 - 73 + 54
B8 184 47104 10111000 - 72 + 55 CLV
B9 185 47360 10111001 - 71 + 56 LDA HHLL,Y
BA 186 47616 10111010 - 70 + 57 TSX
BB 187 47872 10111011 - 69 + 58
BC 188 48128 10111100 - 68 + 59 LDY HHLL,X
BD 189 48384 10111101 - 67 + 60 LDA HHLL,X
BE 190 48640 10111110 - 66 + 61 LDX HHLL,Y
BF 191 48896 10111111 - 65 + 62
C0 192 49152 11000000 - 64 + 63 CPY #DD
C1 193 49408 11000001 - 63 + 64 CMP (LL,X)
C2 194 49664 11000010 - 62 + 65
C3 195 49920 11000011 - 61 + 66
C4 196 50176 11000100 - 60 + 67 CPY LL
C5 197 50432 11000101 - 59 + 68 CMP LL
```

```
C6 198 50688 11000110 - 58 + 69 DEC LL
C7 199 50944 11000111 - 57 + 70
C8 200 51200 11001000 - 56 + 71 INY
C9 201 51456 11001001 - 55 + 72 CMP #DD
CA 202 51712 11001010 - 54 + 73 DEX
CB 203 51968 11001011 - 53 + 74
CC 204 52224 11001100 - 52 + 75 CPY HHLL
CD 205 52480 11001101 - 51 + 76 CMP HHLL
CE 206 52736 11001110 - 50 + 77 DEC HHLL
CF 207 52992 11001111 - 49 + 78 .
D0 208 53248 11010000 - 48 + 79 BNE DEP
D1 209 53504 11010001 - 47 + 80 CMP (LL),Y
D2 210 53760 11010010 - 46 + 81
D3 211 54016 11010011 - 45 + 82
D4 212 54272 11010100 - 44 + 83
D5 213 54528 11010101 - 43 + 84 CMP LL,X
D6 214 54784 11010110 - 42 + 85 DEC LL,X
D7 215 55040 11010111 - 41 + 86
D8 216 55296 11011000 - 40 + 87 CLD
D9 217 55552 11011001 - 39 + 88 CMP HHLL,Y
DA 218 55808 11011010 - 38 + 89
DB 219 56064 11011011 - 37 + 90
DC 220 56320 11011100 - 36 + 91
DD 221 56576 11011101 - 35 + 92 CMP HHLL,X
DE 222 56832 11011110 - 34 + 93 DEC HHLL,X
DF 223 57088 11011111 - 33 + 94
E0 224 57344 11100000 - 32 + 95 CPX #DD
E1 225 57600 11100001 - 31 + 96 SBC (LL,X)
E2 226 57856 11100010 - 30 + 97
E3 227 58112 11100011 - 29 + 98
E4 228 58368 11100100 - 28 + 99 CPX LL
E5 229 58624 11100101 - 27 +100 SBC LL
E6 230 58880 11100110 - 26 +101 INC LL
E7 231 59136 11100111 - 25 +102
E8 232 59392 11101000 - 24 +103 INX
E9 233 59648 11101001 - 23 +104 SBC #DD
EA 234 59904 11101010 - 22 +105 NOP
EB 235 60160 11101011 - 21 +106
EC 236 60416 11101100 - 20 +107 CPX HHLL
ED 237 60672 11101101 - 19 +108 SBC HHLL
EE 238 60928 11101110 - 18 +109 INC HHLL
EF 239 61184 11101111 - 17 +110
F0 240 61440 11110000 - 16 +111 BEQ DEP
F1 241 61696 11110001 - 15 +112 SBC (LL),Y
```

```
F2 242 61952 11110010 - 14 +113
F3 243 62208 11110011 - 13 +114
F4 244 62464 11110100 - 12 +115
F5 245 62720 11110101 - 11 +116 SBC LL,X
F6 246 62976 11110110 - 10 +117 INC LL,X
F7 247 63232 11110111 -  9 +118
F8 248 63488 11111000 -  8 +119 SED
F9 249 63744 11111001 -  7 +120 SBC HHLL,Y
FA 250 64000 11111010 -  6 +121
FB 251 64256 11111011 -  5 +122
FC 252 64512 11111100 -  4 +123
FD 253 64768 11111101 -  3 +124 SBC HHLL,X
FE 254 65024 11111110 -  2 +125 INC HHLL,X
FF 255 65280 11111111 -  1 +126
```

Les codes d'affichage sont, dans l'ordre:
Hexadecimal,Decimal,PRINT,PLOT,FILL,LIST

```
00   0  NUL   0C---  0000C---   NUL
01   1  SOH   0C--R  0000C--R   SOH
02   2  STX   0C-V-  0000C-V-   STX
03   3  ETX   0C-VR  0000C-VR   ETX
04   4  EOT   0CB--  0000CB--   EOT
05   5  ENQ   0CB-R  0000CB-R   ENQ
06   6  ACK   0CBV-  0000CBV-   ACK
07   7  BEL   0CBVR  0000CBVR   BEL
08   8   BS   0A-0-  0000A-0-    BS
09   9   HT   0A-0H  0000A-0H    HT
0A  10   LF   0A-1-  0000A-1-    LF
0B  11   VT   0A-1H  0000A-1H    VT
0C  12   FF   0AC0-  0000AC0-    FF
0D  13   CR   0AC0H  0000AC0H    CR
0E  14   SO   0AC1-  0000AC1-    SO
0F  15   SI   0AC1H  0000AC1H    SI
10  16  DLE   1C---  0001C---   DLE
11  17  DC1   1C--R  0001C--R   DC1
12  18  DC2   1C-V-  0001C-V-   DC2
13  19  DC3   1C-VR  0001C-VR   DC3
14  20  DC4   1CB--  0001CB--   DC4
15  21  NAK   1CB-R  0001CB-R   NAK
16  22  SYN   1CBV-  0001CBV-   SYN
17  23  ETB   1CBVR  0001CBVR   ETB
```

18	24	CAN	1AT60	0001AT60	CAN
19	25	EM	1AT61	0001AT61	EM
1A	26	SUB	1AT50	0001AT50	SUB
1B	27	ESC	1AT51	0001AT51	ESC
1C	28	FS	1AG60	0001AG60	FS
1D	29	GS	1AG61	0001AG61	GS
1E	30	RS	1AG50	0001AG50	RS
1F	31	US	1AG51	0001AG51	US
20	32			00X	
21	33	!	!	00X X	!
22	34	"	"	00X X	"
23	35	#	#	00X XX	#
24	36	$	$	00X X	$
25	37	%	%	00X X X	%
26	38	&	&	00X XX	&
27	39	'	'	00X XXX	'
28	40	(	(	00X X	(
29	41	)	)	00X X X	)
2A	42	*	*	00X X X	*
2B	43	+	+	00X X XX	+
2C	44	,	,	00X XX	,
2D	45	-	-	00X XX X	-
2E	46	.	.	00X XXX	.
2F	47	/	/	00X XXXX	/
30	48	0	0	00XX	0
31	49	1	1	00XX X	1
32	50	2	2	00XX X	2
33	51	3	3	00XX XX	3
34	52	4	4	00XX X	4
35	53	5	5	00XX X X	5
36	54	6	6	00XX XX	6
37	55	7	7	00XX XXX	7
38	56	8	8	00XXX	8
39	57	9	9	00XXX X	9
3A	58	:	:	00XXX X	:
3B	59	;	;	00XXX XX	;
3C	60	<	<	00XXXX	<
3D	61	=	=	00XXXX X	=
3E	62	>	>	00XXXXX	>
3F	63	?	?	00XXXXXX	?
40	64	à	à	01	à
41	65	A	A	01 X	A
42	66	B	B	01 X	B
43	67	C	C	01 XX	C

44	68	D	D	01 X	D
45	69	E	E	01 X X	E
46	70	F	F	01 XX	F
47	71	G	G	01 XXX	G
48	72	H	H	01 X	H
49	73	I	I	01 X X	I
4A	74	J	J	01 X X	J
4B	75	K	K	01 X XX	K
4C	76	L	L	01 XX	L
4D	77	M	M	01 XX X	M
4E	78	N	N	01 XXX	N
4F	79	O	O	01 XXXX	O
50	80	P	P	01 X	P
51	81	Q	Q	01 X X	Q
52	82	R	R	01 X X	R
53	83	S	S	01 X XX	S
54	84	T	T	01 X X	T
55	85	U	U	01 X X X	U
56	86	V	V	01 X XX	V
57	87	W	W	01 X XXX	W
58	88	X	X	01 XX	X
59	89	Y	Y	01 XX X	Y
5A	90	Z	Z	01 XX X	Z
5B	91	'	'	01 XX XX	'
5C	92	£	£	01 XXX	£
5D	93	§	§	01 XXX X	§
5E	94	^	^	01 XXXX	^
5F	95	_	_	01 XXXXX	_
60	96	'	'	01X	'
61	97	a	a	01X X	a
62	98	b	b	01X X	b
63	99	c	c	01X XX	c
64	100	d	d	01X X	d
65	101	e	e	01X X X	e
66	102	f	f	01X XX	f
67	103	g	g	01X XXX	g
68	104	h	h	01X X	h
69	105	i	i	01X X X	i
6A	106	j	j	01X X X	j
6B	107	k	k	01X X XX	k
6C	108	l	l	01X XX	l
6D	109	m	m	01X XX X	m
6E	110	n	n	01X XXX	n
6F	111	o	o	01X XXXX	o

```
70 112   p      p    Ø1XX        p
71 113   q      q    Ø1XX   X    q
72 114   r      r    Ø1XX  X     r
73 115   s      s    Ø1XX  XX    s
74 116   t      t    Ø1XX X      t
75 117   u      u    Ø1XX X X    u
76 118   v      v    Ø1XX XX     v
77 119   w      w    Ø1XX XXX    w
78 120   x      x    Ø1XXX       x
79 121   y      y    Ø1XXX  X    y
7A 122   z      z    Ø1XXX X     z
7B 123   é      é    Ø1XXX XX    é
7C 124   ù      ù    Ø1XXXX      ù
7D 125   è      è    Ø1XXXX X    è
7E 126   è      è    Ø1XXXXX     è
7F 127              Ø1XXXXXX
80 128 ØC--- I0C--- I000C--- END
81 129 ØC--R I0C--R I000C--R EDIT
82 130 ØC-V- I0C-V- I000C-V- STORE
83 131 ØC-VR I0C-VR I000C-VR RECALL
84 132 ØCB-- I0CB-- I000CB-- TRON
85 133 ØCB-R I0CB-R I000CB-R TROFF
86 134 ØCBV- I0CBV- I000CBV- POP
87 135 ØCBVR I0CBVR I000CBVR PLOT
88 136 ØA-Ø- I0A-Ø- I000A-Ø- PULL
89 137 ØA-ØH I0A-ØH I000A-ØH LORES
8A 138 ØA-1- I0A-1- I000A-1- DOKE
8B 139 ØA-1H I0A-1H I000A-1H REPEAT
8C 140 ØACØ- I0ACØ- I000ACØ- UNTIL
8D 141 ØACØH I0ACØH I000ACØH FOR
8E 142 ØAC1- I0AC1- I000AC1- LLIST
8F 143 ØAC1H I0AC1H I000AC1H LPRINT
90 144 1C--- I1C--- I001C--- NEXT
91 145 1C--R I1C--R I001C--R DATA
92 146 1C-V- I1C-V- I001C-V- INPUT
93 147 1C-VR I1C-VR I001C-VR DIM
94 148 1CB-- I1CB-- I001CB-- CLS
95 149 1CB-R I1CB-R I001CB-R READ
96 150 1CBV- I1CBV- I001CBV- LET
97 151 1CBVR I1CBVR I001CBVR GOTO
98 152 1AT60 I1AT60 I001AT60 RUN
99 153 1AT61 I1AT61 I001AT61 IF
9A 154 1AT50 I1AT50 I001AT50 RESTORE
9B 155 1AT51 I1AT51 I001AT51 GOSUB
```

```
9C 156 1AG60 I1AG60 I001AG60 RETURN
9D 157 1AG61 I1AG61 I001AG61 REM
9E 158 1AG50 I1AG50 I001AG50 HIMEM
9F 159 1AG51 I1AG51 I001AG51 GRAB
A0 160       I      I0X       RELEASE
A1 161   !   I   !  I0X    X  TEXT
A2 162   "   I   "  I0X   X   HIRES
A3 163   #   I   #  I0X   XX  SHOOT
A4 164   $   I   $  I0X  X    EXPLODE
A5 165   %   I   %  I0X  X X  ZAP
A6 166   &   I   &  I0X  XX   PING
A7 167   '   I   '  I0X  XXX  SOUND
A8 168   (   I   (  I0X X     MUSIC
A9 169   )   I   )  I0X X  X  PLAY
AA 170   *   I   *  I0X X X  CURSET
AB 171   +   I   +  I0X X XX  CURMOV
AC 172   ,   I   ,  I0X XX    DRAW
AD 173   -   I   -  I0X XX X  CIRCLE
AE 174   .   I   .  I0X XXX   PATTERN
AF 175   /   I   /  I0X XXXX  FILL
B0 176   0   I   0  I0XX      CHAR
B1 177   1   I   1  I0XX   X  PAPER
B2 178   2   I   2  I0XX  X   INK
B3 179   3   I   3  I0XX  XX  STOP
B4 180   4   I   4  I0XX X    ON
B5 181   5   I   5  I0XX X X  WAIT
B6 182   6   I   6  I0XX XX   CLOAD
B7 183   7   I   7  I0XX XXX  CSAVE
B8 184   8   I   8  I0XXX     DEF
B9 185   9   I   9  I0XXX  X  POKE
BA 186   :   I   :  I0XXX X   PRINT
BB 187   ;   I   ;  I0XXX XX  CONT
BC 188   <   I   <  I0XXXX    LIST
BD 189   =   I   =  I0XXXX X  CLEAR
BE 190   >   I   >  I0XXXXX   GET
BF 191   ?   I   ?  I0XXXXXX  CALL
C0 192   @   I   @  I1        !
C1 193   A   I   A  I1     X  NEW
C2 194   B   I   B  I1    X   TAB(
C3 195   C   I   C  I1    XX  TO
C4 196   D   I   D  I1   X    FN
C5 197   E   I   E  I1   X X  SPC(
C6 198   F   I   F  I1   XX   @
C7 199   G   I   G  I1   XXX  AUTO
```

```
C8 200  H  I  H   I1  X    ELSE
C9 201  I  I  I   I1  X  X THEN
CA 202  J  I  J   I1  X X  NOT
CB 203  K  I  K   I1  X XX STEP
CC 204  L  I  L   I1  XX   +
CD 205  M  I  M   I1  XX X -
CE 206  N  I  N   I1  XXX  *
CF 207  O  I  O   I1  XXXX /
D0 208  P  I  P   I1 X     ^
D1 209  Q  I  Q   I1 X   X AND
D2 210  R  I  R   I1 X  X  OR
D3 211  S  I  S   I1 X  XX >
D4 212  T  I  T   I1 X X   =
D5 213  U  I  U   I1 X X X <
D6 214  V  I  V   I1 X XX  SGN
D7 215  W  I  W   I1 X XXX INT
D8 216  X  I  X   I1 XX    ABS
D9 217  Y  I  Y   I1 XX  X USR
DA 218  Z  I  Z   I1 XX X  FRE
DB 219  [  I  [   I1 XX XX POS
DC 220  \  I  \   I1 XXX   HEX$
DD 221  ]  I  ]   I1 XXX X &
DE 222  ^  I  ^   I1 XXXX  SQR
DF 223  _  I  _   I1 XXXXX RND
E0 224  `  I  `   I1X      LN
E1 225  a  I  a   I1X    X EXP
E2 226  b  I  b   I1X   X  COS
E3 227  c  I  c   I1X   XX SIN
E4 228  d  I  d   I1X  X   TAN
E5 229  e  I  e   I1X  X X ATN
E6 230  f  I  f   I1X  XX  PEEK
E7 231  g  I  g   I1X  XXX DEEK
E8 232  h  I  h   I1X X    LOG
E9 233  i  I  i   I1X X  X LEN
EA 234  j  I  j   I1X X X  STR$
EB 235  k  I  k   I1X X XX VAL
EC 236  l  I  l   I1X XX   ASC
ED 237  m  I  m   I1X XX X CHR$
EE 238  n  I  n   I1X XXX  PI
EF 239  o  I  o   I1X XXXX TRUE
F0 240  p  I  p   I1XX     FALSE
F1 241  q  I  q   I1XX   X KEY$
F2 242  r  I  r   I1XX  X  SCRN
F3 243  s  I  s   I1XX  XX POINT
```

```
F4 244   t   I  t   I1XX X   LEFT$
F5 245   u   I  u   I1XX X X RIGHT$
F6 246   v   I  v   I1XX XX  MID$
F7 247   w   I  w   I1XX XXX
F8 248   x   I  x   I1XXX
F9 249   y   I  y   I1XXX  X
FA 250   z   I  z   I1XXX X
FB 251   é   I  é   I1XXX XX
FC 252   ù   I  ù   I1XXXX
FD 253   è   I  è   I1XXXX X
FE 254   è   I  è   I1XXXXX
FF 255     I    I1XXXXXX
```

B) LA ROM PAR THEME

1-Saisie de paramètres

Evaluation d'expressions

CA98 CAE2 EVALUER UN NUMERO DE LIGNE EN #33-#34

E79D E853 EVALUER UN ENTIER NON SIGNE DANS YA,#33-#34 ET #D4-#D3

CE77 CF03 EVALUER UNE EXPRESSION NUMERIQUE DANS ACC1

D20A D29C PRENDRE UN ENTIER NON SIGNE DANS #D4-#D3

CE7A CF06 VERIFIER VARIABLE NUMERIQUE

CE7C CF08 VERIFIER VARIABLE CHAINE

CE8B CF17 EVALUER UNE EXPRESSION DANS ACC1

CF74 CE00 RAMASSER UN OPERANDE DANS ACC1

D0FC D188 PRENDRE ADRESSE D'UNE VARIABLE

D186 D216 TESTER SI A EST ALPHABETIQUE

D80A D8C5 SAUTER UN CARACTERE ET PRENDRE UN ENTIER DANS

D30D D8C8 PRENDRE UN ENTIER DANS X

D85B D916 PRENDRE DEUX PARAMETRES ENTIERS

D996 DA22 PRENDRE DEUX COORDONNES (PLOT)

DA6B DAF6 INTERDIRE MODE HIRES

D419 D4D2 INTERDIRE MODE DIRECT

E725 E7B2 SYNTAXE CSAVE/LOAD

Gestion des séparateurs

CFD3 D05F DEMANDER UNE ')'

CFD6 D062 DEMANDER UNE '('

CFD9 D065 DEMANDER UNE ','

CFDB D067 DEMANDER A

2-Routines d'interprétation

Gestion de TXTPTR

00E2 00E2 LIRE CARACTERE SUIVANT

00E8 00E8 LIRE CARACTERE COURANT

C765 C73A PLACER TXTPTR AU DEBUT DU TEXTE BASIC

CA0D CA3F AJOUTER Y A TXTPTR

CA1C CA4E CALCUL DANS Y DU DEPLACEMENT A LA PROCHAINE INSTRUCTION

CA1F CA51 CALCUL DANS Y DU DEPLACEMENT A LA FIN DE LA LIGNE

D352 D90D RECUPERER TXTPTR DANS #E0-#E1

Autres routines

C000 C000 RESET BASIC

C003 C003 ENTREE INTERPRETEUR

C3CA C3C6 RECHERCHE DE BLOCS BASIC SUR LA PILE

C3F8 C3F4 DECALER UN BLOC MEMOIRE VERS LE HAUT

C432 C437 VERIFIER PLACE SUR LA PILE

C448 C444 VERIFIER AY EN DESSOUS DES CHAINES

C4B5 C4A8 ENTREE INTERPRETEUR

C56F C55F RESTAURER LES LIENS DES LIGNES

C5A2 C592 PRENDRE UN ORDRE DANS LE TAMPON CLAVIER

C5F8 C5E8 ATTENDRE UN CARACTERE AU CLAVIER

C60A C5FA ENCODAGE DES MOTS CLES

C6DE C6B3 RECHERCHE D'UNE LIGNE BASIC

C719 C6EE 'NEW' COMMANDE

C738 C70D 'CLEAR' COMMANDE

C91F C952 'RESTORE' COMMANDE

3-Conversions et transferts de nombres

Conversions numériques

DF15 DF24 A --) ACC1 (SIGNE)

D3ED D499 YA --) ACC1 (SIGNE)

D8D5 DF40 YA --) ACC1 (NON SIGNE)

D217 D2A9 ACC1 --) ENTIER NON SIGNE (#D4-#D3)

D867 D922 ACC1 --) YA (NON SIGNE)

DEEC DEF4 AACC1 --) ACC1

DF74 DF8C ACC1 --> #D4-3-2-1 (SIGNE)

E9D1 E9D5 ACC1 --> DECIMAL #100 ...

Transferts de nombres

DE73 DE7B (AY) --> ACC1

DD4D DD51 (AY) --> ACC2

DECD DED5 ACC2 --> ACC1

DEE9 DEE8 ACC1 --> ACC2

DEDD DEE5 AACC1 --> ACC2

DE98 DEA9 AACC1 --> #CB-#CF

DE9B DEA3 AACC1 --> #C6-#CA

DEA1 DEA9 AACC1 --> (#00B8)

DEA5 DEAD AACC1 --> XY-XY+4

DEA8 DEB9 ACC1 --> XY-XY+4

Quelques constantes (stockées en flottant)

DC6F DCA5 -0,5

E40D E411 0,25

E201 E205 0,5

DC4B DC81 1

DDBA DDBE 10

E9A7 E9AB 999.999,9

E9AC E9B9 999.999.999

E9B1 E9B5 1.000.000.000

DC6A DCAØ SQR(2)

D9E9 DC7C PI

E4Ø3 E4Ø7 PI/2

E4Ø9 E4ØC 2*PI

DC74 DCAA LN(2)

E278 E27C 1/LN(2)

DC46 DC77 LN(1Ø)

La gestion des chaines

D4FØ D5AB RESERVER UNE CHAINE DE LONGUEUR A

D595 D65Ø REORGANISER LES CHAINES

D6AC D767 AJOUTER DEUX CHAINES

D6F7 D7B2 TRANSFERT DE CHAINES

D712 D7CD PRENDRE DESCRIPTEUR ET VERIFIER CHAINE

D715 D7DØ PRENDRE DESCRIPTEUR

4-Routines de calculs

Calculs divers

DA79 DBØ4 ACC1+Ø,5 --> ACC1

EØ72 EØ76 ACC1+A --> ACC1

DD7A DD7E ACC1*2^A --> ACC1

DDA3 DDA7 ACC1*1Ø --> ACC1

DDBF DDC3 ACC1/1Ø --> ACC1

DEFC DFØB ACC1=Ø

DFØØ DFØF ACC1=-1

DF04 DF13 A=SIGNE DE ACC1

E26D E271 ACC1=-ACC1

E2F9 E2FD CALCULER UN POLYNOME DEGRE IMPAIR,INDEXE PAR AY

E3ØF E313 CALCULER UN POLYNOME INDEXE PAR AY

Opérateurs

Remarque: (*) signifie que A,N et Z doivent être représentatifs de l'exposant de ACC1 (donc un LDA DØ doit précéder l'appel à ces opérateurs), ceci étant inutile dans tous les autres cas.

DØ87 D113 ')=(' OPERATEUR

DA8Ø DBØB (AY)-ACC1 --) ACC1

DA83 DBØE ACC2-ACC1 --) ACC1

DA97 DB22 (AY)+ACC1 --) ACC1

DA9A DB25 ACC2+ACC1 --) ACC1 (*)

DCB7 DCED (AY)*ACC1 --) ACC1

DCBA DCFØ ACC2*ACC1 --) ACC1 (*)

DDEØ DDE4 (AY)/ACC1 --) ACC1

DDE3 DDE7 ACC2/ACC1 --) ACC1 (*)

E234 E238 ACC2^ACC1 --) ACC1 (*)

Fonctions

Remarque: le paramètre et le résultat sont dans ACC1

DC79 DCAF 'LN' FONCTION

DDDØ DDD4 'LOG' FONCTION

DF31 DF49 'ABS' FONCTION

DFA5 DFBD 'INT' FONCTION

E22A E22E 'SQR' FONCTION

E2A6 E2AA 'EXP' FONCTION

E34B E34F 'RND' FONCTION

E387 E38B 'COS' FONCTION

E38E E392 'SIN' FONCTION

E3D7 E3DB 'TAN' FONCTION

E43B E43F 'ATN' FONCTION

Calculs simples

F7ØØ F731 AY=4Ø%A

D965 DAØC CALCULER ADRESSE LIGNE A (MODE TEXT)

EEFF EFC8 DIVISION ENTIERE

5-Affichage et impression

.... C816 IMPRIMANTE EN SERVICE

.... C82F IMPRIMANTE HORS SERVICE

CB9F CBFØ ALLER A LA LIGNE

CBED CCBØ AFFICHER LA CHAINE POINTEE PAR AY

CCØA CCCE EFFACER L'ECRAN TEXTE

CC12 CCD9 AFFICHER A SUR LE TERMINAL COURANT

F73F F77C AFFICHER X SUR L'ECRAN

F57B F5C1 AFFICHER A SUR L'IMPRIMANTE

F5D3 F602 AFFICHER LE CARACTERE DE CONTROLE A SUR L'ECRAN

E0B6 E0BA AFFICHER 'IN' ET LE NUMERO DE LIGNE

E0C1 E0C5 AFFICHER XA COMME ENTIER NON SIGNE

E563 E5F5 EFFACER LIGNE 0

F82F F865 ECRIRE SUR LA LIGNE 0

6-Graphique

EDE3 EED8 PLACER FB CODE

F250 F2E4 TESTER PARAMETRE NON NUL ET ASSEZ PETIT

F264 F2F8 TESTER PARAMETRE ASSEZ PETIT

F2C3 F35D SAUVER MOTIF ET ADRESSE POINT COURANT

F2D4 F36E RECUPERER MOTIF ET ADRESSE POINT COURANT

EFA6 F049 CALCULER ADRESSE DU POINT (X,Y)

EFE6 F089 DEPLACER POINT VERS LE BAS

EFF5 F095 DEPLACER POINT VERS LE HAUT

F004 F0A1 DEPLACER POINT VERS LA DROITE

F015 F0B2 DEPLACER POINT VERS LA GAUCHE

EDF6 EEE8 AFFICHER UN POINT (X,Y)

EF4D F016 AFFICHER POINT COURANT

F141 F1C8 EVALUER 'COULEUR' D'UN POINT

EE06 EEF9 DRAW CALCUL ET AFFICHAGE

F331 F3C6 CIRCLE CALCUL ET AFFICHAGE

F0EB F171 CHAR: CALCULER ADRESSE DU CARACTERE

F115 F19B CHAR: AFFICHER LE CARACTERE

F17F F204 PAPER

F18B F210 INK

7-Son

F535 F590 ROUTINE D'ENTREE/SORTIE AVEC LE PSG 8912

FA6C FA86 ENVOYER 14 PARAMETRES AU PSG 8912

FA85 FA9F PING

FA9B FAB5 SHOOT

FAB1 FACB EXPLODE

FAC7 FAE1 ZAP

FAFA FB14 SON CLAVIER 1

FB10 FB2A SON CLAVIER 2

FB26 FB40 SOUND

FBB6 FBD0 PLAY

FC00 FC1A MUSIC

8-Gestion des cassettes

E5C6 E65E ECRIRE UN OCTET SUR LA CASSETTE

E630 E6C9 LIRE UN OCTET SUR LA CASSETTE

E696 E735 TROUVER LA BANDE AMORCE

E6BA E75A ECRIRE LA BANDE AMORCE

E6CA E76A CONFIGURER VIA POUR TRAVAIL CASSETTE

E804 E93D RECONFIGURER VIA

.... E4AC PRENDRE ENTETE DU PROGRAMME

.... E607 SAUVER ENTETE DU PROGRAMME

.... E4E0 CHARGER/VERIFIER LE PROGRAMME

.... E62E SAUVER LE PROGRAMME

E4A8 .... PRENDRE ENTETE ET CHARGER PROGRAMME

E57B .... SAUVER ENTETE ET SAUVER LE PROGRAMME

E554 E56C PASSER OCTET SUIVANT, TEST DE FIN

9-Routines systèmes

Les interruptions

ECC7 EDE0 AUTORISER IRQ PAR T1

ED01 EE1A INTERDIRE IRQ PAR T1

0228 0244 VECTEUR IRQ

023D 024A VECTEUR RETOUR IRQ

ED09 EE22 GESTION NORMALE DES IRQ

022B 0247 VECTEUR NMI

F882 F8B2 GESTION NORMALE DES NMI

F888 F8B8 CONFIGURER LE SYSTEME

F960 F9AA INITIALISER LE VIA 6522

Les timers

ED70 EE8C ANNULER LES TROIS TIMERS

ED81 EE9D PRENDRE VALEUR DANS YX DU TIMER A

ED8F EEAB INITIALISER LE TIMER A AVEC LA VALEUR YX

EDAD EEC9 INITIALISER TIMER ET ATTENDRE QU'IL S'ANNULLE

configuration écran

D93D D9E4 LORES

E9A9 EC21 TEXT

E9BB EC33 HIRES

ED8C .... EFFACER ECRAN HIRES

F7E0 F816 GENERER LES CARACTERES ALTERNES

Gestion du clavier

E905 EB78 GERER LE TAMPON TOUCHE

F43C F495 GERER LE CLAVIER

Gestion de la mémoire

E965 EBD8 HIMEM

E974 EBE7 GRAB

E994 EC0C RELEASE

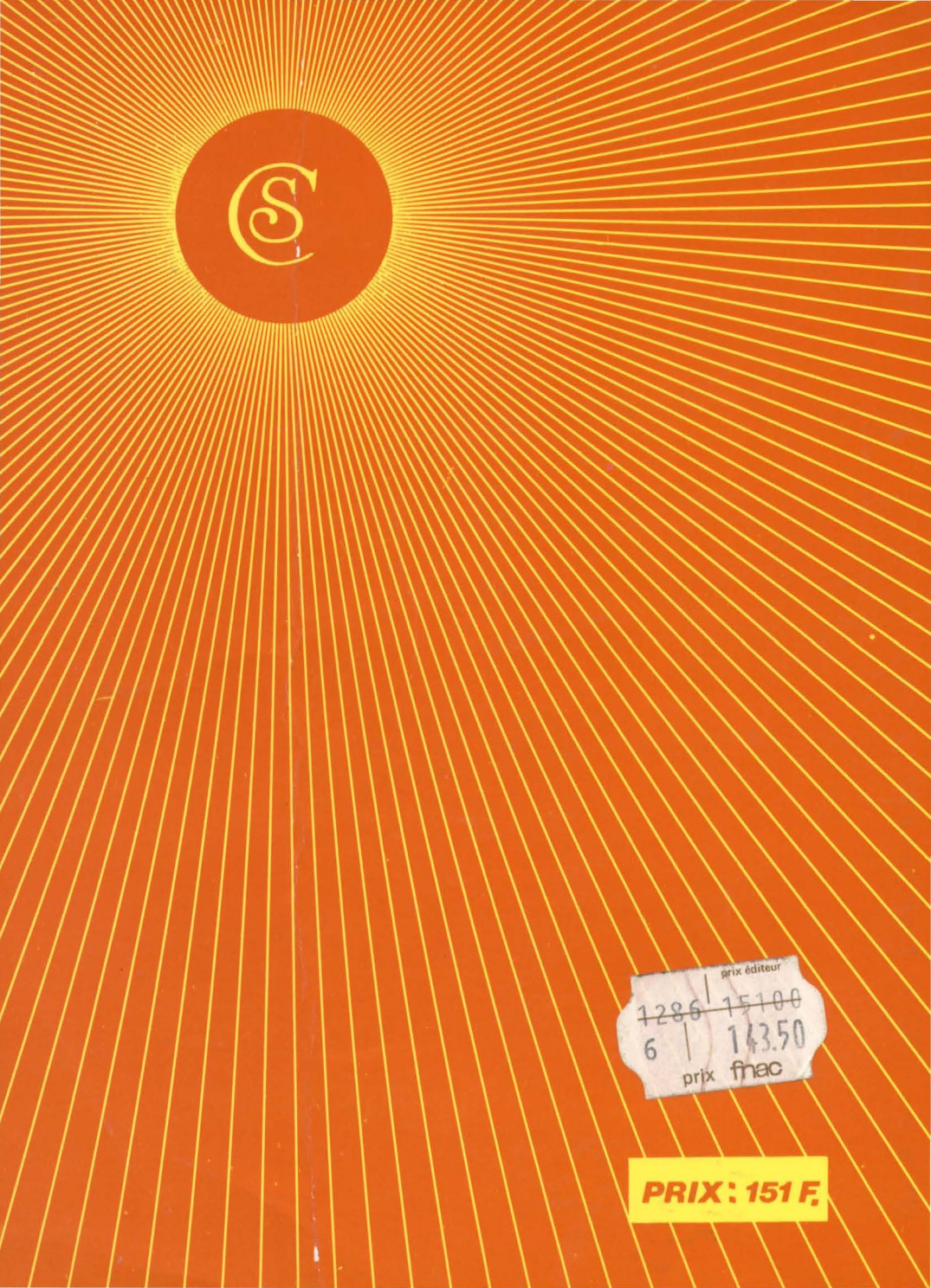
SOMMAIRE

6-Les routines graphiques
7-Les routines sonores
8-La gestion des cassettes
9-Les routines systèmes

Composition : Par l'Auteur
Maquette : SORACOM
Impression : JOUVE
No d'Editeur: 051

ISBN 2904032 - 40 -1

Dépôt légal : Juillet 1986


prix éditeur
1286 | 15100
6 | 143.50
prix fnac
PRIX : 151 F.