Ce dossier est strictement reservé aux membres de PARADOX. Ne pas le diffuser a des personnes etrangéres au groupe S.V.P.

I) LE CRTC. CATHODE RAY TUBE CONTROLLER.

RAMLAID

1) Les registres.

Les registres du CRTC sont au nombre de 18, mais pour les ruptures, seuls 10 registres sont utiles (les registres 0,1,2,3,4,6,7,9,12 et 13). Vous connaissez tres bien les registres 1,2,6,7,12 et 13, mais les registres 0,3,4 et 9 meritent quelques explications.

a) Le registre 4.

Ce registre defini le nombre de ligne de caractere que doit balayer le canon a electrons. Cette valeur est diminuée de 1 par rapport au nombre réel de ligne de caractere a l'ecran. Par exemple, quand vous allumez le CPC, le registre 4 est a 38, et le nombre de ligne de caractere a l'ecran est de 39. on a donc : R4=nb de ligne de caractere - 1
La valeur du reg 4 peut varier de 0 a 127. A 128, le CRTC boucle, c'est a dire qu'il revient a 0. Par exemple, si vous mettez 128 dans le reg 4, le CRTC conciderera le 128 comme un 0. Si vous mettez 138, il conciderera le 138 comme un 10, etc...

b) Le registre 9.

Ce registre defini le nombre de ligne de pixel qu'il y a dans un caractere. Il est egalement diminué de 1 par rapport au nombre réel de ligne par caractére. Reg 9= nb ligne par caractere - 1.
La valeur normale est de 7 (conc 8 lignes par caractére (7+1)) cette valeur peut varier de 0 a 31. A 32, le CRTC boucle.

c) Le registre 3.

Il defini la largeur de la HBL (Horizontal Blanking), c'est la bande noire qu'il y a tout a gauche de l'ecran (on ne peut pas la voir, car elle est trop a gauche) Cette largeur est definie en MOT (1 MOT=2 OCTETs) elle varie de 0 (donc 0 octets) a 15 (donc 30 octets). A 16, ca boucle. La valeur normale est 14. La HBL sert au canon a electrons a se synchroni-ser. Lorsque la valeur du R3 est trop basse, le canon se synchronise mal et l'ecran est decalé (ce bug est utilisé pour faire des scrolls hards a l'octet, mais le resultat n'est pas tres propre, car le scroll vibre un peu, car le canon n'est pas bien synchronisé! Si la valeur est inferieure a 4, le canon est si mal synchronisé que l'ecran n'est plus stable.

d) Le registre 0.

Ah, le registre 0. Il est utilisé depuis peu de temps par les demo-makers en effet, son fonctionnement et surtout son utilité etait assez vague. Ce registre correspond en fait au reg 4, mais en horizontal. Il defini le nombre de MOT qu'il y a sur une ligne. Sa valeur normale est 63. Il est lui aussi decrementé par rapport au nombre réel de MOT par ligne. Donc, si le R0 est a 63, il y a 64 MOTs sur une ligne, ce qui fait 128 octets. Sa valeur varie de 0 a 255.

2) Le fonctionnement des registres.

a) Les registres 12 et 13.

Ces registres definissent l'ofset de depart de la memoire ecran. Cette adresse est codée sur 10 Bits, de 0 a 9. Les bits 10 et 11 sont utilisés

pour la taille de l'ecran (16 ou 32 Ko), et les bits 12 et 13 gerent la zone video utilisée. Ces 2 registres sont les seuls a pouvoir etre bufferisé. C'est a dire que vous pouvez les charger avant que l'ecran ne commence a etre affiché. Si vous faites une rupture sur 2 ecrans, vous pouvez changer l'ofset du deuxieme ecran, pendant que le premier est affiché. Le changement des reg 12 et 13 ne sera prit en compte que lorsque le deuxieme ecran commencera.

b) Les registres compteurs.

Les registres 0,3,4 et 9 sont des compteurs. Lorsque le CRTC affiche un ecran, il fait varier quelque part dans ses petits circuits,la valeur de chacun de ces registres. Il possede pour chacun de ces registres, une sorte de variable. Lorsque l'on change la valeur d'un reg, on change le maximum de la variable correspondante. Par exemple, le reg 4 est a 38 (sa valeur normale), quand le CRTC commence a afficher l'ecran, la variable est a 0, et a chaque fois qu'il a fini d'afficher une ligne de caractere, il incremente la variable, jusqu'a ce qu'elle atteigne son maximum, qui est ici 38. A 38, l'ecran a fini d'etre affiché, et il commence un nouvel ecran. Idem pour le registre 9, quand le CRTC commence un caractere, la variable est a 0, et a chaque fin de ligne, il incremente le compteur, jusqu'a 7. A 7, le compteur recommence a 0.

c) L'overflow.

L'overflow est un comportement anormal du CRTC du au fait que l'on envoie une valeur incorecte dans un registre. L'overflow le plus connu est celui du reg 7, qui permet de coller les ecrans dans une rupture. En effet, on met le reg 7 a 255, qui est une valeur overflow, et de ce fait, le CRTC ne produit plus de VBL (qui est la bande noire tout en haut de l'ecran, on peut la voir en mettant le border a une autre valeur que 0, et en tournant doucement le bouton qu'il y a derriere le moniteur. Cette bande permet au canon de se resynchroniser en debut d'affichage d'un ecran.) Mais il y a d'autres overflows. Lorsque vous modifier la valeur d'un registre compteur, il peut se produire un overflow. Par exemple, vous mettez le reg 4 a 20, mais au moment ou vous mettez cette valeur, la variable du reg 4 en est a 25. Dans ce cas, il se produit un overflow. En effet, vous fixer le maximum du reg 4 a 20, mais le compteur en est a 25, donc, avant d'arriver a 20, il va aller jusqu'a son maximum (qui est 127) vous allez donc avoir un ecran qui defile, ou qui scintille.
Ceci est valable pour tous les registres compteurs. Vous avez peut-etre constaté un overflow en faisant de la rupture par ligne. Si vous avez en debut de rupture ligne a ligne, 4 lignes de caracteres identiques, cela est du a un overflow. Vous mettez le reg 9 a 0, alors que le compteur en est par exemple a 2. Dans le cas, le reg 9 va a son maximum, qui est 31 (32 lignes de pixels, ca fait bien 4 lignes de caracteres.)

3) Les types.

Les CRTCs sont classés en 5 types. Les types 0,1 et 2 sont pour les CPCs vielle generation. Le CRTC 2 est le plus difficil a maitriser, en effet, sa structure est beaucoup moins souple que les autres CRTCs. Pour le programmer en rupture normale, il faut respecter de nombreuses regles. En ruptures plus modernes (ligne a ligne, verticales, etc...) il est presque impossible d'y parvenir, car il plante lorsqu'on lui demande des choses trop poussés (Si l'on fait par exemple R4=0 et R9=0, ca plante.) Bref, c'est un CRTC de merde, si vous en avez un, il ne vous reste plus qu'une solution : LE SUICIDE.
Le CRTC 3 est celui du CPC+, et le 4 est un type quasiment inexistant. Depuis peu de temps, LONGSHOT et OVERFLOW ont créé un nouveau type de classement. En effet, l'ancien système n'était pas très fiable. Sur tous

les CRTC 0 ne reagissaient pas de la meme facon, idem pour les autres types. Donc une nouvelle classification s'imposait. Maintenant, il n'y a plus que 2 types, type A et type B.Ces 2 types regroupes tous les anciens types, sauf les types 2 (enfin, les vraix types 2, car certain CRTC repondent type 2 au test de LONGSHOT, mais sont en realité des types 1 ou 0.) Voici a quoi on reconnait un type A d'un type B :

Type A : - scroll transparent par REG 6
- RVI non buggée
- RVMB non buggée

Type B : - scroll transparent par REG 8
- RVI buggée
- RVMB buggée

Pour la signification de RVI et RVMB, voir plus loin. Si vous voulez savoir si vous avez un CRTC A ou B, lancer la demo NEW AGE 1. Si le message : PRESS A apparait, vous avez un CRTC type B, si le message PRESS B apparait, vous avez un CRTC type A. C'est logique tout ca !

4) Les regles.

Lorsque l'on commence a modifier les registres CRTC de facon assez barbare, il faut respecter certaines regles, si l'on ne veut pas se retrouver avec un plantage.
Tout d'abord il faut que le registre 7 soit inferieur ou egal au registre 4. Mais cette regle n'est qu'a appliquée lorsque le canon se trouve en bas d'un ecran, et qu'il va generer une VBL. En effet, lorsque dans une rupture vous mettez le reg 7 a 255, il n'y a pas plantage car en fin de balayage, vous remettez le 7 a une valeur normale.
Il faut egalement que le reg 2 soit inferieur ou egal au registre 0. cette regle doit etre observée dans tous les cas.
Sur les CRTC 2, il faut que la somme des registres 2 et 3 soit inferieure ou egale au registre 0.
Le registre 3 doit etre supperieur ou egal a 4, car sinon, le canon n'a pas le temps de se synchroniser, et l'ecran n'est plus stable.
Il faut egalement que lors d'une rupture, il y ai au total, un nombre proche de 312 lignes de pixels par ecran. Avec les CRTC 2, il faut etre tres tres proche de 312, les autres accepte une marge plus grande d'erreur. Le nombre de ligne est donné par les registres 4,5 et 9.
Bon, voici un resumé en formules des conditions, avec en plus, 2 ou 3 autres conditions :

reg 7 <= reg 4
reg 2 <= reg 0 (CRTC 2 : reg 2+reg3 (= reg 0)
reg 3 >=4
(1) (reg4+1)*(reg9+1)+reg5=312 (avec marge +ou- grande suivant CRTC.)
reg 6 <= reg 7
reg 1 <= reg 2

Pour la formule (1), dans une rupture, il faut faire la somme de toutes les lignes pour chaque rupture, et cette somme doit etre egale a 312. Si vous avez une rupture a X ecrans, voici ce que ca donne :

rupture 1 rupture 2 rupture X
(reg4+1)*(reg9+1)+reg5 + (reg4+1)*(reg9+1)+reg5 +...+ (reg4+1)*(reg9+1)+(reg5) =312.

Oui, je sais, ca parrait peut etre un peu compliqué, mais en fait, c'est tout simple. Il faut toujours garder a l'esprit que l'on doit avoir un nombre de ligne proche de 312.

II) La temporisation.

Dans les ruptures, la temporisation est tres importante. Dans une rupture classique, la tempo n'est pas tres difficil a gerer car elle se fait en general a l'aide des HALTs. Mais dans une rupture ligne a ligne, il faut etre synchro à la ligne près, et dans les ruptures verticales, il faut etre synchro au NOP près ! Il est donc fondamental de savoir calculer le

temps machine.
Tout d'abord, il est utile de savoir que les HALTs interviennent toutes les 52 lignes. Il y a 6 HALTs par balayage, donc cela fait bien 312 lignes au total (6*52). Si on modifie la longueur d'une ligne (avec le registre 0) les HALTs ne seront plus toutes les 52 lignes. Si vous mettez le R0 a 31 (donc, a la moitié de sa valeur) vous aurez des HALTs toutes les 26 lignes. En effet, sur une ligne réelle d'ecran, on aura 2 lignes pour le CTRC. Si vous ne comprennez pas tres bien, ce n'est pas grave, je reviendrais sur ce sujet a propos des ruptures verticales.
Maintenant, venons en au calcul de temps machine. Si vous regardez les tables de temps machine qu'il y a sur certain livres, vous verez des temps en micro secondes, ou en cycles d'horloge. Tout cela n'est pas pratique. Je vous donnerais plus loin un tableau avec les temps en NOPs. Le NOP etant une des instructions les plus rapides, je me suis basé sur le NOP pour calculer le temps machine des autres instructions.
Sur une ligne d'ecran, il y a l'equivalent de 64 NOPs (lorsque R0=63). Si vous faites un raster, il faut que votre routine qui boucle ait une longueur d'execution de 64 NOPs, sinon le raster sera decalé.
Je me suis rendu compte que les tables de temps machine de la plupart des livres etaient foireuses. En effet, les temps donnés etaient completement faux. Certain livres allaient meme jusqu'a donner des temps machines a virgules ! En effet, voici un exemple : LD A,n 7 micro seconde, ce qui fait 1,75 NOPs. Ceci est totalement debil car le Z80 fonctionne en tranches de 4 cycles (donc 1 NOP), il ne peut pas executer la fin d'une instruction, et le debut d'une autre sur la même tranche de 4 cycles.
Bon, voici une table de temps machine que j'ai calculée moi-meme (grace a un raster.)

R=registre 8 BITs (A,B,C,D,E,H,L) DD=REGISTRE 16 BITs (BC,DE,HL)
n=nombre 8 BITs nn=nombre 16 BITs
cc=condition (Z, NZ, C, NC, etc...) XY=REGISTRE IX ou IY
Les durées sont exprimées en NOP.

LD R,R	1	LD R,n	2	LD R,(HL)	2
LD R,(XY+n)	5	LD (HL),R	2	LD (XY+n),R	5
LD (HL),n	3	LD (XY+n),n	6	LD A,(DD)	2
LD A,(nn)	4	LD (DD),A	2	LD (nn),A	4
LD A,I	3	LD A,R	3	LD I,A	3
LD R,A	3	LD DD,nn	3	LD XY,nn	4
LD HL,(nn)	5	LD DD,(nn)	6	LD XY,(nn)	6
LD (nn),HL	5	LD (nn),DD	6	LD (nn),XY	6
LD SP,HL	2	LD SP,XY	3	PUSH DD	4
PUSH XY	5	POP DD	3	POP XY	4
EX DE,HL	1	EX AF,AF'	1	EXX	1
EX (SP),HL	6	EX (SP),XY	7	LDI	5
LDIR	5+6*(BC-1)	LDD	5	CPI	4
LDDR	5+6*(BC-1)	CPIR	4+6*(BC-1)	CPD	4
CPDR	4+6*(BC-1)	ADD A,R	1	ADD A,n	2
ADD A,(HL)	2	ADD A,(XY+n)	5	INC R	1
INC (HL)	3	INC (XY+n)	6	DAA	1
CPL	1	NEG	2	CCF	1
SCF	1	NOP	1	HALT	1
DI	1	EI	1	IM	2
ADD HL,DD	3	ADD HL,XY	4	ADC HL,DD	4
ADC HL,DD	4	ADD XY,DD	4	INC DD	2
INC XY	3	RLCA	1	RLA	1
RRCA	1	RRA	1	RLC R	2
RLC (HL)	4	RLC (XY+n)	7	RLD	5
RRD	5	BIT n,R	2	BIT n,(HL)	4
BIT n,(XY+n)	6	SET n,R	2	SET n,(HL)	3
SET n,(XY+n)	7	JP nn	3	JP cc,nn	3
JR n	3	JR cc,n	3 si cc vraie et 2 si cc fausse.		

JP (HL)	1	JP (XY)	2 ?	CALL nn	5
RET	3	DJNZ n	3 si b different de 1, 2 sinon.		
RETI	4	CALL cc,nn	3 si cc vraie et 2 sinon.		
RETN	4	IN A,(n)	3	IN R,(C)	4
INI	5	IND	5	OUT (n),A	3
OUT (C),R	4	OUTI	5	OUTD	5
ADC A,R	1	ADC A,n	2	ADC A,(HL)	2
ADC A,(XY+n)	5	SUB R	1	SUB n	2
SUB (HL)	2	SUB (XY+n)	5	SBC A,R	1
SBC A,n	2	SBC A,(HL)	2	SBC A,(XY+n)	5
AND/OR/XOR R	1	AND/OR/XOR n	2	AND/OR/XOR (HL)	2
AND/OR/XOR (XY+n)	5	CP R	1	CP n	2
CP (HL)	2	CP (XY+n)	5	DEC R	1
DEC (XY+n)	5	RL/RR R	2	RL/RR (HL)	4
RL/RR (XY+n)	7	SLA/SRA R	2	SLA/SRA (HL)	4
SLA/SRA (XY+n)	7	RRC/SRL R	2	RRC/SRL (HL)	4
RRC/SRL (XY+n)	7	RES n,R	2	RES n,(HL)	4
RES n,(XY+n)	7				

Ouf ! Voila, vous pouvez vous fiez a cette table a 100%, elle est tout a fait correcte (a moins d'avoir fait une erreur de frappe.)
Allez, on fait un petit exercice de calcul de temps machine :

```
BOUC    LD A,(HL)      2
        INC A          1
        ADD A,5        2
        SLA A          2
        LD (HL),A      2
        OUT (C),A      4
        INC HL         2
        DJNZ BOUC      3 et 2 quand B sera a 1 (en fin de boucle)
            total: 18 et 17 en fin de boucle.
```

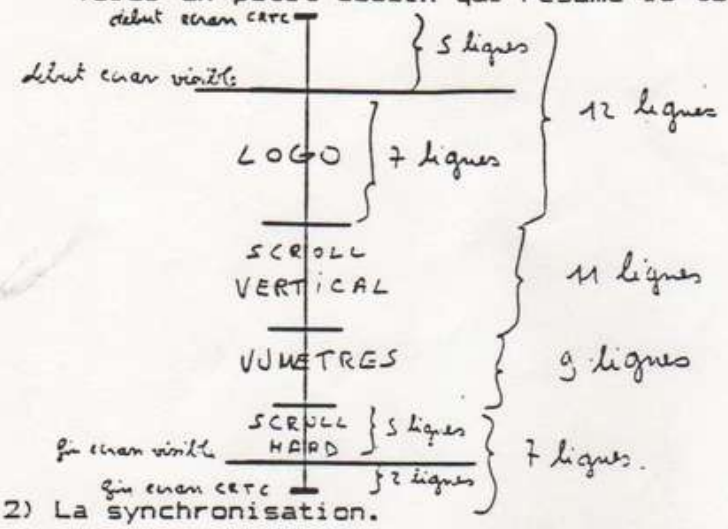
Je ne sais pas du tout a quoi peut servir cette boucle, je l'ai tapée sans reflechir, c'est juste un exemple.

II) La rupture classique.

1) Le format.

Lorsque l'on veut faire une rupture, il faut tout d'abord definir son format, c'est a dire le nombre d'ecrans et leur hauteur. Il faut ensuite transformer ces hauteurs en valeurs a envoyer au CRTC (il suffit de diminuer la hauteur de 1.) Prenons par exemple une rupture avec 4 ecrans. La somme des hauteurs (en caractere) des ecrans doit etre 39.
Sur le moniteur nous voyons a peut pres 32 lignes caracteres. Il y a encore 2 lignes caracteres en dessous, et 5 lignes caracteres au dessus (ceci n'est valable que dans la structure que je vous propose. Cette structure est la plus simple pour reussir n'importe quelle rupture.)
Bon, dans notre rupture, on veut qu'il y ai en bas, un scroll hard de 5 lignes de caracteres. Ceci nous fait un ecran de 5 lignes de haut, tout en bas, mais etant donnée qu'il y a encore 2 lignes caracteres en dessous on va ajouter 2 lignes a cet ecran, pour avoir ainsi un ecran en bas de 7 lignes (5 seront vues, et les 2 dernieres seront trop basses.) on veut mettre tout en haut un logo sur un ecran de 7 lignes de haut. Il faut ajouter 5 lignes car il y en a 5 qui ne sont pas visibles au dessus, ceci nous fait donc un total de 7+5=12 lignes caracteres en haut. Hein ? Kwa ? Vous ne comprenez rien ? Attendez, je ferais un dessin plus loin pour que vous compreniez mieux. Bon, on a donc un ecran en bas de 7 lignes, et un en haut de 12 lignes. On veut mettre en dessous du logo un scroll vertical de 11 lignes de haut, on a donc un ecran de 11 lignes. Et enfin en dessous du scroll vertical, un ecran fixe, pour mettre des vumetres par exemple. Cet ecran sera de ? lignes de haut. Voila, verifions que le

total fait 39 : 12+11+9+7=39, pas de probleme. Les valeurs a envoyer au CRTC sont : 11 (12-1), 10 (11-1), 8 (9-1), et 6 (7-1).
Voici un petit dessin qui resume le tout.



2) La synchronisation.

Pour effectuer les changement des reg 4, il faut etre bien synchro, pour cela on utilisera les HALTs. Pour que tout cela soit clair, on fait un petit dessins, sur lequel on represente l'ecran. Chaque case corresponc a une ligne de caractere. On place les HALTs (toutes les 52 lignes de pixels, donc, toutes les 6,5 lignes de caractere (52/8=6,5)) On dessine nos ecrans en fonction de leur taille, et on obtient ainsi la position des changements d'ecran par rapport aux HALTs. Regarder plus loin, il y a ce dessin et aussi des dessins d'autres exemples. (Feuille quadrillée 1)

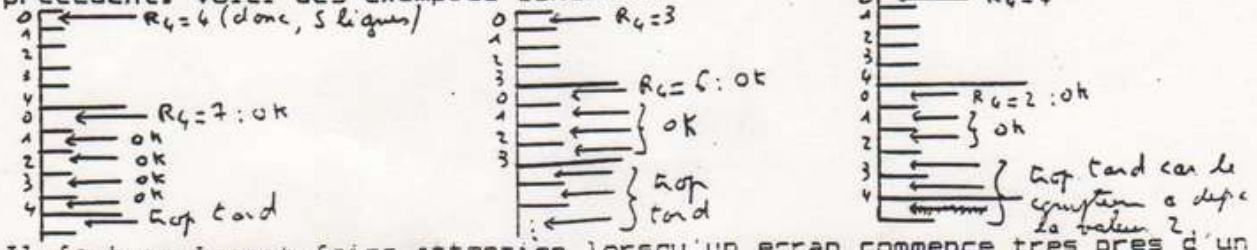
3) Le changement des registres.

Tout d'abord, pour que nos ecrans soit collés, on met au debut du balayage, le reg 7 a 255, ce registre est en overflow, donc les ecrans seront collés. Apres le BD19, on met une petite boucle pour etre certain que le CRTC ait commencer d'afficher l'ecran. Maintenant, on peut commencer a faire nos ruptures. Apres la boucle, on fixe le maximum du R4 a 11, pour lui indiquer la dimension de notre premier ecran. Apres le 1er HALT, nous seront toujours sur le premier ecran. Apres le 2eme HALTs, nous seront sur le 2eme ecran, donc, on peut changer le R4 pour cet ecran et on le met a 10, ainsi on defini la taille du 2eme ecran. Apres le 3eme HALT nous seront toujours sur le 2eme ecran. Apres le 4eme HALT, nous seront sur le 3eme ecran, on change donc le R4 pour definir notre 3eme ecran. On met le R4 a 8. Apres le 5eme HALT, nous seront sur le dernier ecran, on change le R4, et on le met a 6. Mais attention, nous sommes la dans le cas particulier ou le debut de l'ecran se situe a moins de 1 caractere avant le HALT, et dans ce cas, il faut mettre une petite tempo avant de changer le R4 et le R7. Voila, notre rupture est faite, ce n'est pas plus dificil que ca. Il reste juste une derniere chose a faire, c'est mettre le R7 a une valeur non OVERFLOW, car sinon l'ecran ne sera pas stable. Donc, apres le dernier HALT, on met le R7 a 0. Je vous conseille de toujours mettre le R7 a 0 et pas une autre valeur, en fin de balayage, cela permet d'appliquer ce que je viens de vous exposer. Si vous mettez une autre valeur, votre shema de l'ecran sera different.

4) Attention a l'overflow.

Dans l'exemple que j'ai pris, j'ai fait attention qu'il n'y ai pas de probleme d'OVERFLOW lors du changement du registre 4, mais lorsque vous ferez vos ruptures, il faudra faire attention a ce probleme. Pour eviter

ce probleme, il faut que lorsque vous modifiez le registre 4, il se soit ecoulé un nombre de ligne inferieur ou egal au nombre de ligne de l'ecran precedent. Voici des exemples concrets.



Il faut egalement faire attention lorsqu'un ecran commence tres pres d'un HALT (exemple, les ruptures 1 et 5 de la feuille quadrillée. Le dernier ecran de ces ruptures commence tres pres du 5eme HALT.) Dans ce cas, il est preferable de mettre une petite tempo avant de changer le registre 4. Ceci pour eviter les problemes de compatibilité. En effet, d'un CRTC a l'autre, il y a de petites variations quand a l'arrivé des HALTs, donc. lorsqu'un ecran commence a moins d'une ligne de caractere d'un HALT, mettez une tempo (voyez dans les exemples sur le disk.)
Il y a donc sur le disk, 6 exemples de ruptures, pour illustrer cette partie. Ces ruptures sont detaillées sur la feuille quadrillée.
Apres ces explications, vous etes normalement capable de realiser n'importe quelle rupture classique.

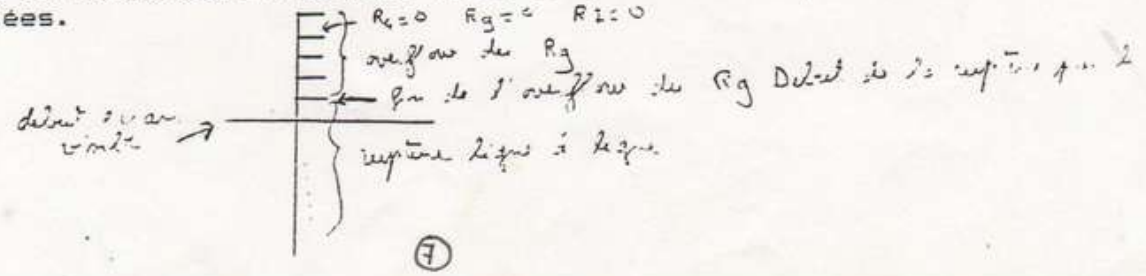
IV) La rupture ligne a ligne.

1) Le principe.

Le principe de la rupture ligne a ligne est tres simple. Il suffit de mettre les registres 4 et 9 a 0, et mettre le R7 a 0 ou 255, suivant les cas. On met le R7 a 0, si on commence la rupture ligne a ligne aussitot apres le FRAME. Si apres cette rupture par ligne, on met d'autres ecrans. il faut mettre le R7 a 255, car on retourne dans le cas de ruptures classiques. Si vous faites une rupture ligne a ligne apres un ecran, il faut mettre le R7 a 255, comme pour une rupture classique. En resumé, juste apres le FRAME : R7=0. Apres un ecran : R7=255.
Pour le CRTC, chaque ligne representera un ecran. En effet, le registre 4 represente la taille en hauteur d'un ecran, celui-ci etant a 0 l'ecran fait pour le CRTC 1 caractere de haut. Le registre 9 est a 0, donc chaque caractere fait 1 ligne de haut, donc chaque ecran fait 1 ligne de haut !
Une fois que vous avez modifié les registres 4, 9 et 7, et s'il n'y a pas d'overflow, la rupture ligne a ligne commence, et elle se fait toute seule, vous n'avez plus besoin de modifier les registres 4,7 et 9, vous n'avez qu'a changer les registres 12 et 13.

2) Ah, l'overflow !

Eh oui, pour la rupture ligne a ligne, il faut respecter 2 overflows, celui du registre 4 et celui du registre 9. Pour eviter tout probleme, on peut faire les modifications en tout debut de balayage, ainsi le registre 4 en est encore a 0, donc il n'y a pas d'overflow de ce registre. Par contre le registre 9 ne sera certainement pas a 0 donc il y aura overflow. Mais cela n'est pas un probleme car l'overflow du registre 9 produit un caractere de 32 lignes avant de faire des caracteres de 1 ligne, et avant d'arriver au debut de l'ecran visible, le canon parcour 5 lignes de caracteres (ce qui fait 40 lignes de pixels) donc, en arrivant au debut de l'ecran visible, l'overflow sera terminé, et le reg 9 aura un comportement normal. Voici un petit dessin pour vous eclairsir les idées.



Si vous voulez faire votre rupture ligne a ligne au milieu de l'ecran, vous ne pouvez pas utiliser cette methode, puisque vous devrez mettre les reg 4 et 9 a zero en milieu d'ecran, et dans ce cas on verait l'overflow de registre 9. Il faut donc bien synchroniser le changement des reg 4 et 9. Cette synchronisation se fait a la ligne pres ! Il faut encore travailler sur papier pour bien voir ce que l'on fait (voir feuille quadrillée numero 2.) Le seul moyen est tout d'abord de reperer en gros ou se situe le changement des reg 4 et 9, et de faire votre routine en fonction de cette aproximation. Avec cette aproximation vous avez 7 chances sur 8 d'avoir un overflow du reg 9, donc pour le supprimer, vous travaillez par tatonement, en ajoutant ou en supprimant 64 NOPs (64 NOPs representent le temps que met le canon a parcourir une ligne.)

3) La gestion et l'affichage d'une rupture par ligne.

Eh oui, maintenant que vous savez couper l'ecran toutes les lignes, il faut savoir modifier l'adresse toutes les lignes. Pour cela, on utilise toujours les registres 12 et 13. Comme je l'ai dit precedement, vous n'avez plus a modifier les registres 4, 7 et 9, il suffit de changer les 12 et 13, mais pas n'importe comment, il faut une bonne synchro, et il faut savoir gerer les adresses de chaque ligne. Pour la gestion des adresses, je vous conseille de faire un tableau des adresses de chaque ligne, et vous changez ce tableau a tous les balayages. Ensuite, lors de l'affichage, vous allez chercher l'adresse, et l'envoyer dans les reg 12 et 13. Ensuite, il faut attendre d'etre sur la ligne suivante, il faut bien synchroniser votre routine, pour changer les 12 et 13 toujours au meme moment sur les lignes, il faut faire comme si c'etait un raster. La routine peut ressembler a cela par exemple :

```
            HL points sur la table des adresses (valeur R12 puis R13)
            A = nombre de lignes
            BC = #BC0C
      BOUC  OUT (C),C  On selectionne le reg 12
            LD D,(HL)  On prend la valeur du R12
            INC HL     On augmente la position de lecture dans la table
            INC B      B=#BD
            OUT (C),D  On envoie la valeur du R12
            INC C      On augmente C, qui contient maintenant 13
            DEC B      B=#BC
            OUT (C),C  Selectionne le reg 13
            LD D,(HL)  Prend la valeur du R13
            INC HL
            INC B      B=#BD
            OUT (C),D  Envoie la valeur du R13
            DEC C      Remet C a 12
            DEC B      B=#BC
            DEFS 30,0  correspond a 30 NOPs, pour la synchronisation
            DEC A
            JR NZ,BOUC
```

La boucle fait 64 NOPs de temps machine, donc la synchronisation est parfaite, les registres 12 et 13 seront changés toujours au meme moment sur chaque ligne. Pour modifier les registres 12 et 13, vous pouvez aller plus vite en utilisant l'instruction OUTI, mais ici, ce n'est pas utile. Attention, l'instruction OUTI est un peu speciale, je vous en parlerai plus loin.

4) Les avantages, les inconvenients.

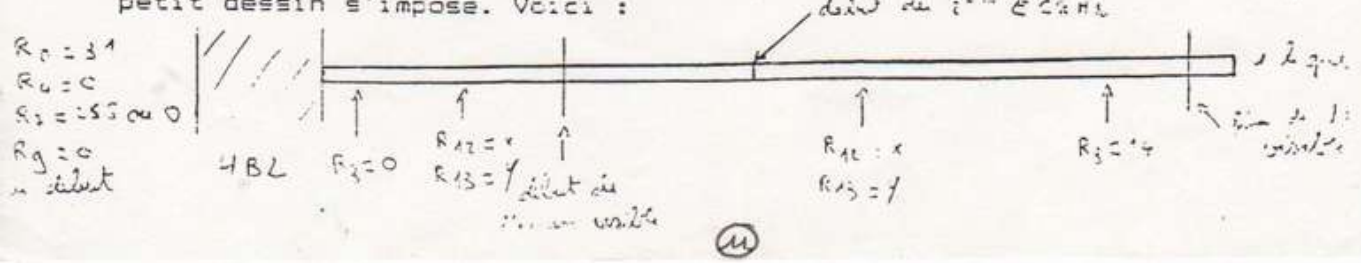
Le principal avantage de la rupture ligne a ligne, est de pouvoir changer l'ofset de l'ecran a chaque ligne (c'est d'ailleur pour cela que la rupture ligne a ligne a été inventée.) Cela permet par exemple de faire se deformer un logo en donnant l'impression qu'il tourne autour d'un axe

(dans la partie de PICT dans THE DEMO, et egalement dans la PHOENIX PART de LONGSHOT toujours dans THE DEMO.) Cela permet egalement de faire rebondir plusieurs scrolls texts identiques en meme temps (dans la partie de SLASH dans FUCKING EXAM) Il y a encore plein d'autre choses que l'on peut faire avec une rupture par ligne.
Mais le principal inconvenient de ce genre de rupture est qu'elle est gourmande en place memoire video, pour des effets peu spectaculaires. En effet, une ligne representant un ecran, et etant donné que par les reg 12 et 13 vous ne pouvez adresser que le debut d'une zone memoire (Vous ne pouvez adresser que les zones #0000-#07FF, #4000-#47FF, #8000-#87FF et #C000-#C7FF) vous avez sur un espace de #4000 octets, seulement #800 octets de memoire video, ce qui represente 25 lignes de pixel (pour des lignes de 80 octets.) Cela est du au fait que vous ne pouvez pas mettre comme ofset de debut d'ecran des valeurs telles que #C800, #D000... Si vous utilisez toute la memoire, vous n'aurez que 200 lignes de pixels, et vous ne pourez plus faire d'autres ecrans, ou scrolls hards, car une zone utilisée par une rupture par ligne ne peut plus etre utilisée pour un autre type de rupture, sinon, vous auriez en debut de chaque ligne de caractere, ce qu'il y a dans votre rupture par ligne. Par contre, vous pouvez dans cette zone, mettre des DATAs, des musiques, du prog, à condition de le mettre en dehors de la memoire adressable par les reg 12 et 13, c'est a dire, entre #0800 et #3FFF, #4800 et #7FFF, #8800 et #BFFF et entre #C800 et #FFFF. En resumé, si dans la zone #4000-#7FFF (par exemple) vous faire une rupture par ligne, vous aurez entre #4000 et #47FF le contenu de la rupture par ligne, et vous pourrez mettre autre chose entre #4800 et #7FFF (datas, musiques, prog, etc...) mais vous ne pourrez plus faire autre chose de cette zone (comme par exemple un scroll hard, ou un ecran fixe.)
Pour remedier au fait que l'on ne peut pas adresser toute la memoire a travers les reg 12 et 13, OVERFLOW, a inventé la RVI, et ce que j'ai batisé la RVMP, mais je vous en parlerais plus loin.

V) La rupture verticale.

1) Le principe.

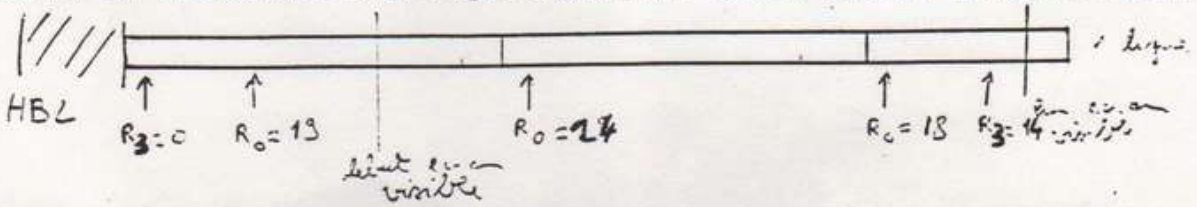
La rupture verticale a eté inventée par LONGSHOT, mais sa technique n'etait pas tres interressante, c'est OVERFLOW qui a réelement fait la premiere rupture verticale UTILISABLE.
La rupture verticale n'est rien d'autre d'une extention de la rupture par ligne. Nous savons maintenant couper l'ecran toutes les lignes. Mais si on retrecissait la longueur d'une ligne ? Comment me direz-vous ? Eh bien avec le reg 0. Le registre 0 defini la longueur d'une ligne en mot (1 mot = 2 octets.) Sa valeur normale est 63 (donc 64 MOTs, ce qui fait 128 octets.) si on mettait le reg 0 a 31 (donc 32 MOTs, soit 64 octets.) la ligne serait coupée en 2. Aie, en plein milieu d'une ligne il y a une grosse HBL, beurk, que c'est laid. Pour la supprimer, il fait mettre le registre 3 a 0. Voila, il n'y a plus de HLB, mais la rupture n'est pas tres droite. Nous avons maintenant une rupture verticale telle que LONGSHOT l'a decouverte il y a quelques temps (decembre 1990 exactement.) mais la rupture n'etant pas droite, on ne peut pas l'utiliser. Pourquoi la rupture n'est pas droite ? Eh bien car le registre 3 est a 0, donc, en debut de ligne, le canon n'a pas le temps de se synchroniser, et l'ecran n'est pas stable. Il a fallu attendre decembre 1991, pour que OVERFLOW (encore lui !) ai l'idée de changer le R3 pendant l'affichage de la ligne afin que le canon puisse se synchroniser. Donc, son idée est de mettre le R3 a 0 en debut de ligne, afin de ne pas avoir de HBL entre les morceaux d'ecrans sur une ligne, et de mettre le R3 a 14 (ou une valeur comprise entre 4 et 15) en fin de ligne pour que le canon se synchronise. Bon, un petit dessin s'impose. Voici :



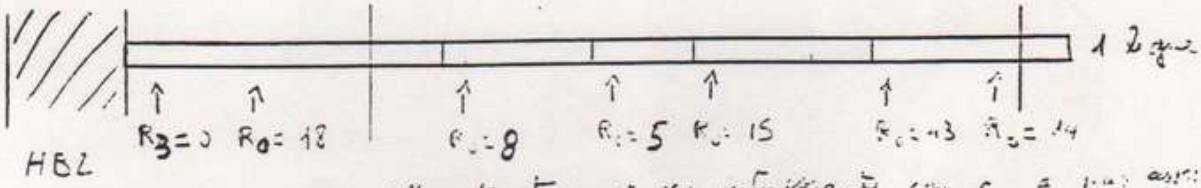
Et voila une rupture verticale parfaite ! Cool non ?

2) D'autres Ruptures Verticales.

Nous ne somme pas limité a 2 ecrans par ligne (eh oui, car maintenant, chaque ecran ne fait plus 1 ligne, mais une demi ligne, ou meme moins !) nous pouvons en mettre 4,8,16,32, etc... Mais disons que au dessus de 4 ecrans par ligne, il devient difficil de gerer correctement la rupture. Pour avoir 4 ecrans par ligne, il faut des lignes 4 fois plus petite, donc, 128/4=32. 32 octets font 16 mots. Il faut donc mettre dans le reg 0 la valeur 15 (16-1). Si vous voulez 8 ecrans par ligne, il faut mettre la valeur 7 (8-1) dans le reg 0. Vous pouvez egalement faire 3,5 ou 6 ecrans mais pour cela, il faut changer le registre 0 a chaque ligne. Imaginons que l'on veut avoir 3 ecrans sur une ligne, le premier de 40 octets, le suivant de **50** octets, et le dernier de **38** octets (40+**50**+**38**=128. Il faut absolument que le total des longueurs fasse 128 octets.) Pour cela, il faut proceder comme pour une rupture normale, mais cette fois, en changeant le registre 0, 3 fois par ligne. En debut de ligne, on met dans R0 la valeur **19** (**20** octets font **20** mots, donc on met **19**=**20**-**1**) Ensuite, il faut changer une deuxieme fois le R0, pour le 2 eme ecran. Mais attention a l'overflow, il faut attendre que le 2eme ecran ai commencé d'etre affiché, avant de changer le R0. Idem pour le 3 eme ecran de la ligne. Ensuite, on recommence a la ligne suivante. Allez, encore un petit dessin



Si vous voulez faire une rupture verticale a 5 ecrans, vous pouvez le faire de la meme maniere. Voici un petit exemple :



3) Petites choses a savoir.

Lorsque vous faites une rupture verticale, vous pouvez recentrer les ecrans avec le reg 2, comme s'il n'y avait pas de rupture, mais il faut respecter la regle : R2<R0. Si vous voulez changer le mode etre les ecrans d'une ligne (oui, oui, c'est possible, d'ailleur, je pense le faire pour ma partie de OBSESSION.) ce qui corespond donc a changer le mode sur une meme ligne, il faut que entre les ecrans de la ligne, il y ai une petite HBL (R3>=2) si R3<2 on ne peut pas changer le mode.
Mais dans ce cas, attention, en effet, si dans votre rupture verticale, vous ne mettez pas le R3 à 0 en début de ligne, mais à une autre valeur, vous ne pourrez plus centrer les ecrans avec le R2.
Pour gerer votre rupture, je vous conseille d'utiliser le meme principe qu'avec la rupture par ligne, c'est a dire de faire un tableau, et de faire une routine qui fait exactement 64 Nops. D'ailleur, une bonne synchro est indispensable, car il faut etre synchro au NOP près ! Et oui, c'est dur. De plus, si votre rupture est tres complexe, supprimez la boucle pour gagner du temps machine (vous gagnerez 4 NOPs, ou plus encore suivant votre routine) et copier la routine autant de fois qu'il y a de lignes.

pour aller encore plus vite dans le changement des registres, vous pouvez utiliser l'instruction OUTI. Mais attention, cette instruction decremente le registre B, AVANT d'effectuer le OUT, donc, si vous voulez envoyer le contenu de HL dans le port #BD, vous devrez faire :

```
LD B,#BE
OUTI
```

L'instruction OUTI remplace cela :

```
DEC B
OUT (C),(HL) (Qui n'existe pas)
INC HL
```

Tout cela en 5 NOPs, c'est super rapide non ? C'est surtout tres utile lorsque vous avez un tableau des ofsets ecrans. Vous faites pointer HL sur cette table des ofsets, et vous utilisez OUTI.
Dans une rupture verticale, il faut encore faire attention a une chose, c'est le fait qu'en changeant le R0, on change l'arrivé des HALTs. En effet, les HALTs arrivent toutes les 52 lignes, mais pour compter les fins de ligne, le CRTC regarde le R0. Si on met le R0 a 31 (donc 2 ecrans par ligne), pour le CRTC, il compte 2 lignes pour 1. Donc, les HALTs arriveront 2 fois plus souvent, c'est a dire, toutes les 26 lignes. Si vous avez X ecrans par ligne, les HALTs arriveront toutes les 52/X lignes
Allez, je vous donne encore quelques petits trucs, qui peuvent servir. Lorsque vous voulez envoyer une valeur sur les ports #BC ou #BD, vous pouvez egalement utiliser les ports #0C et #0D, ou encore, les ports #8C et #8D. A quoi cela sert-il ? Ca sert a gagner des registres. Ben oui, pour selectionner le registre 12, au lieu de : LD BC,#BC0C OUT (C),C vous pouvez faire : LD B,#0C OUT (C),B. Et #8C et #8D, ca sert a quoi ? Ca sert pour les changements de mode. Vous avez la valeur pour le port du CRTC, et la valeur a envoyer pour changer le mode. Tout ceci pour economiser des registres.
Dans vos boucles, vous ne pouvez pas employer le registre B (car il contient la valeur du port), donc vous pouvez utiliser le registre A. Mais si dans votre boucle, le registre A est modifié, au lieu de faire un PUSH AF en debut de boucle, et un POP AF en fin de boucle, vous pouvez mettre en debut et en fin de boucle : EX AF,AF'. Ce truc est assez connu, mais je le donne, au cas ou vous ne le connaitriez pas encore. Pour utiliser ceci, il faut couper les interruptions, sinon vous allez perdre la valeur de A.

3) Aventages, inconvenients.

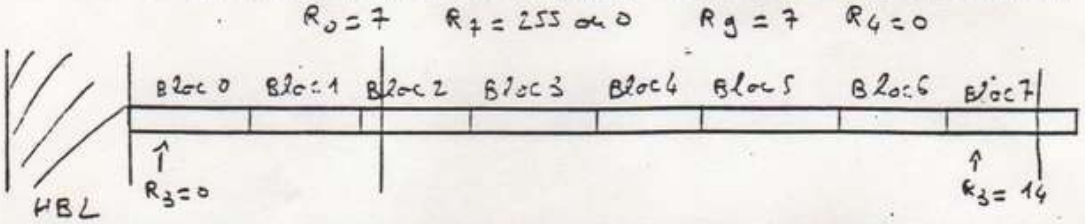
Les inconvenients sont les même que pour une rupture ligne à ligne, c'est à dire la nécessité d'une grande place mémoire, et le gaspillage de mémoire video. Les aventages sont de pouvoir changer l'offset de départ de l'écran, plusieurs fois par ligne. Cela permet de faire des choses du genre de l'intro de S&KOH, je n'ai pas d'autres exemples, car personne d'autre n'a encore fait de rupture verticale (si, moi, mais je n'ai pas encore sorti de demo qui l'utilise, il y a aussi les allemands du groupe SKB qui s'amusent bien avec la rupture verticale, et je crois que c'est tout.) C'est une technique toute nouvelle, qui mérite que l'on s'y interesse.

RVMB, RVI

Bon, la, on entre dans les choses vraiment serieuses. Tout ce qu'il y avait avant, c'etait de la rigolade. RVMB, et surtout RVI sont les techniques les plus recentes. Une seule demo a ete realisée avec la RVI, c'est la fameuse S&KOH. Quand a la RVMB, aucune demo ne l'a encore utilisée, car elle est peut utilisable.
Ces 2 techniques ont pour but de faire une rupture ligne a ligne, ou l'on puisse mettre n'importe quelle zone memoire a l'ecran. Avec une rupture ligne a ligne classique, nous sommes limite au #800 premiers octets d'une zone. Avec la RVMB et la RVI, toute la memoire peut etre utilisée.

1) La Rupture Verticale Multi-Bloc (RVMB)

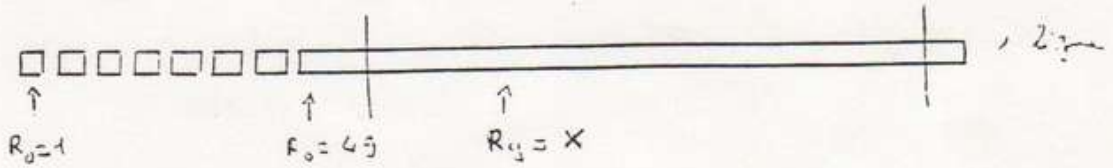
J'appelle par BLOC, l'une des 8 zones de #800 octets qu'il y a sur un ecran. Ce sont les 8 zones qui correspondent aux 8 lignes des caracteres. Sur un ecran en #C000, les 8 blocs sont : #C000-#C7FF, #C800-#CFFF, #D000-#D7FF, #D800-#DFFF, #E000-#E7FF, #E800-#EFFF, #F000-F7FF et #F800-#FFFF. Je vous rapelle qu'avec une rupture ligne a ligne classique, on ne peut utiliser que le premier bloc (#C000-#C7FF).
La RVMB consiste a faire une rupture verticale de 8 ecrans. Chaque ecran est un bloc different. Pour cela, on met le R0 a 7, le R4 a 0, et le R9 a 7. Ensuite, on jere la rupture verticale en mettant le R3 a 0 en debut de ligne, et en le mettant a 14 en fin de ligne. Et on gere les changement d'ofset comme pour une rupture ligne a ligne classique. Etant donné que le R9 est a 7, et que le R0 est a 7, l'ofset ne pourra changer d'une fois par ligne (pour qu'il change plusieur fois, il faut que R9=0), et les 8 ecrans de la ligne auront une largeur de 16 octets. On obtient donc ceci



Avec une ligne a ligne normale, on utilise a chaque ligne, 80 octets (si R1=40) ou plus, du premier bloc (le seul adressable.) Ici, on utilise 16 octets, le reste de la ligne se situe sur les autres blocs. Donc, avec une RVMB, on peut faire 128 lignes differentes de rupture avec une zone de #4000 octets (au lieu de 25 avec une ligne a ligne normale.)
Mais il y a un gros probleme, c'est que cette technique ne fonctionne pas sur les CRTC type B (ca fait a peu pres 1 CPC sur 2 ou 3) donc, on ne peut pas utiliser cette technique si l'on veut une compatibilitee totale. En effet, sur les CRTC type B, on a beau mettre le R3 a 0 en debut de ligne, il y a tout de meme une petite HBL entre les ecrans d'une ligne. Donc, avec une RVMB, la ligne est coupée par ces HBL. Cette remarque est donc egalement a prendre en compte pour les ruptures verticales classiques. En effet, il faut faire attention que les bords de vos ecrans seront legerement 'rognés' sur type B (il y a a peu pres 1 octet de HBL entre les ecrans.)

2) La RVI.

Bon, je n'ai pas encore experimenté cette technique, donc je vous en parle de facon theorique.
Dans la RVI, il faut faire une rupture verticale que l'on ne voit pas ! En effet, la rupture verticale se trouve tres a gauche, ainsi on ne la voit pas sur le moniteur. On met au depart le R4 a 0 et le R7 a 255 (pour avoir une rupture ligne a ligne.) et le R2 a 0 (pour cacher les ruptures verticales.) Ensuite, on fait la rupture verticale. En debut de ligne, on met le R0 a 1, (on ne met pas le R3 a 0, ca ne sert a rien puisqu'on ne voit pas les ruptures verticales.) on attend un peu, afin qu'il se soit affiché 7 ecrans, et on met le R0 a 49 (on a bien 64 : (49+1) + 7*(1+1) = 64.) ensuite on modifie le R9 afin d'avoir le bloc qui nous interesse a l'ecran. Il faut egalement changer les R12 et R13 bien sur. En clair, ca donne ceci :



Donc, on peut changer les R12 et R13 comme l'on veut, mais on est sur le meme bloc me direz-vous. Eh bien non, car comme je l'ai dit, on change a chaque ligne le R9, afin d'avoir le bloc que l'on veut. Allez, un exemple. A l'ecran, on a le bloc 0, on veut a la ligne suivante le bloc 1, on met dans le R9 la valeur 5. A la ligne suivante, il y a 7 petit ecrans. Le premier ecran sera sur le bloc 0, le deuxieme sur le bloc 1, le troisieme sur le bloc 2, le quatrieme sur le bloc 3, le cinquieme sur le bloc 4, et le sixieme sur le bloc 5. Comme on a fixe le R9 a 5, a l'ecran suivant, les blocs repartent a 0. Donc, le septieme ecran sera sur le bloc 0, et enfin, le dernier ecran, celui que l'on a a l'ecran, sera sur le bloc 1. Si a la ligne suivante vous voulez le bloc 2 vous mettez dans le R9 la valeur 5. A la ligne suivante, le premier ecran sera sur le bloc 2 (car a la ligne precedente, le derniere ecran, celui qui est vu, etait sur le bloc 1.), le deuxieme sur le bloc 3, le 3eme sur le bloc 4, le quatrieme que le bloc 5, le cinquieme sur le bloc 0 (car on a mit le R9 a 5) le sixieme a 1, le septieme a 2, et le dernier, celui qui est a l'ecran, est sur le bloc 3.
Voila le principe de la RVI, c'est tres compliqué a comprendre, mais une fois qu'on a compris (j'ai compris, mais je n'ai pas encore mit en pratique.) on peut faire ca les yeux fermés.
Bon, comme ci se n'etait pas assez compliqué, il y a encore des problemes de compatibilité sur type B. En effet, sur le type B, le premier ecran de la ligne n'est pas pris en compte par le CRTC, donc, il faut faire comme s'il n'y avait que 6 ecrans avant celui que l'on voit. Ce qui fait une routine tout a fait differente.
De plus, il y a des combinaisons que l'on ne peut pas avoir. En effet, on ne peut pas avoir sur une ligne un bloc 1, et sur la ligne du dessous, un bloc 6 (par exemple.)
Bon, voila, je ne m'etendrais pas plus sur le sujet car je ne le connais pas encore assez. Si cela vous interresse vraiment, je ferais peut etre plus tard, un dossier complet sur la RVI (une fois que je l'aurais mis en pratique.)
Il a bien du courrage OVERFLOW d'avoir fait ca dans sa demo ! Et il a bien du merite d'avoir inventé cette technique, car c'est vraiment tordu.

VII) Adaptation CRTCs.

Un mot rapide sur les différences entres les CRTCs, afin d'avoir des routines 100% compatibles avec tous les CRTCs.
Pour la rupture classique, si vous utilisez ma méthode, et si vous mettez bien des tempos quand il en faut (quand un écran commence près d'un HALT votre rupture sera compatible avec tous les CRTCs (même les types 2 !)
Si vous faite une rupture ligne à ligne, pour passer d'un CRTC à l'autre il suffit d'ajouter ou d'enlever 64 NOPs avant le changement des R4 et R9 (dans les exemples de rupture ligne à ligne qu'il y a sur le disk, j'ai mit une boucle avant le changement des R4 et R9. La boucle fait 64 NOPs (il y a DEFS 61,0, qui remplace 61 NOPs, et le DJNZ, qui fait 3 NOPs de temps machine, donc 61+3=64 NOPs.) Donc, s'il y a l'overflow du R9, changer la valeur de B (il suffit normalement de l'augmenter ou de la diminuer de 1.)
Et pour les ruptures verticales, il faut augmenter ou diminuer de 1 ligne (comme pour une rupture ligne a ligne) et également, augmenter ou diminuer que quelques NOPs.
Ces changements de temporisation sont identiques pour tous les CRTCs du même type. Il vous suffit de tester si pour passer du CRTC 0 au CRTC 1, il faut ajouter ou enlever 64 NOPs, etc...

VIII) THE END.

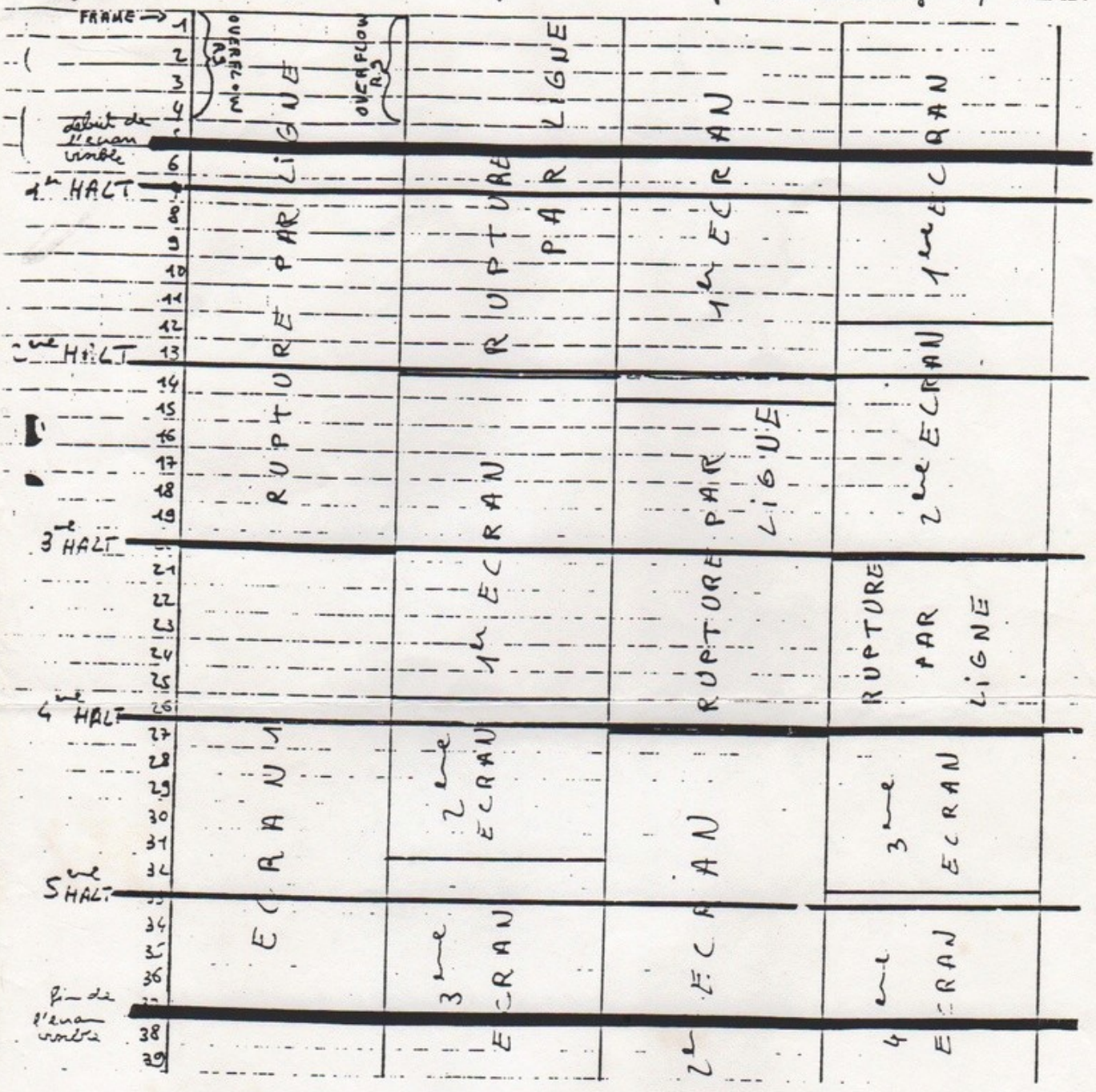
Et voilà, c'est fini ! Je vous ai dit tout ce que je savais. J'espère avoir été clair. J'espère que vous aurez compris, et que cela vous aura

aidé. Excusez les fautes d'orthographe et de style !
Je vous quitte en vous disant que TOUTES les ruptures sont realisable. Il suffit de bien poser le probleme, de faire un beau petit dessin, et de respecter ce que je vous ai dit.
Si vous avez un probleme, si vous voulez un reseignement supplementaire, contacter moi. Bye bye...

GOZEUR.

LOTTIAUX Renaud
9 Rue du moulin
59990 MARESCHES

Tel : 27-49-30-85



RUPTURE LIGNE A LIGNE