AMSTRAD CPC CRTC COMPENDIUM

LOGON SYSTEM

SOMMAIRE

1 PRÉAMBULE

L'objectif de ce document est d'apporter des informations détaillées sur le fonctionnement des différents circuits CRTC 6845 implémentés dans les CPC créés par AMSTRAD. Le CRTC est un circuit contrôleur capable de fournir une interface entre des micro-ordinateurs et des écrans cathodiques gérant un balayage vidéo.

Ce document aborde également le fonctionnement de quelques circuits associés aux CRTC, notamment le GATE ARRAY.

Ces informations peuvent servir à tout programmeur de jeu ou de démo travaillant encore sur ces machines conçues durant les années 1980-1990.
Elles peuvent aussi servir de référence à toute personne souhaitant créer un émulateur autrement qu'en adaptant son code au coup par coup pour des programmes spécifiques.

Enfin, il peut apporter des informations utiles aux utilisateurs d'autres machines équipées de CRTC similaires. Il faut cependant considérer que les timings entre le processeur, le CRTC et le circuit vidéo associé d'une autre machine peuvent modifier notablement les comportements décrits ici. Par ailleurs, ce document décrit uniquement les fonctions des CRTC utilisées sur le CPC. Le mode « texte » et le « curseur » ne sont donc pas traités.

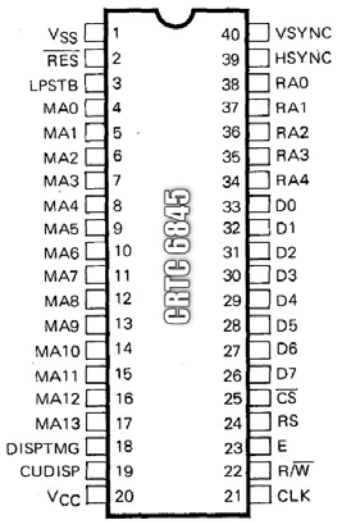
Une partie des informations indiquées dans ce document a été vérifiée à partir d'un programme de benchmark appelé SHAKER. Il a été testé pour chaque CRTC sur plusieurs machines.

Un autre document annexe contient, pour différents tests, les écrans obtenus avec l'émulateur WINAPE avec les photos d'écran pour les CRTC connus à ce jour sur CPC. Lorsqu'il n'y a pas de photo, c'est que l'émulateur n'est pas en mesure d'émuler le CRTC sans perte de synchronisation.

La vérité tient peu de place, mais l'erreur occupe une infinité de lieux.
Ce document est amené à être corrigé et à évoluer.

Serge Querné
Longshot / Logon System
serge.querne@logonsystem.fr

Remerciements



Arnaud STORQ (*NoRecess*) pour son infinie patience et ses talents cachés de cameraman pour m'envoyer les résultats de SHAKER lorsque mes CPC étaient inaccessibles.

Claire DUPAS (*Cheshirecat*) pour son retour d'expérience sur les HSYNC et ses idées parfois pas si farfelues que ça :-)

Relecture gourmande :
Edouard BERGÉ (*Roudoudou*)

Relecture molle :
Sébastien BROUDIN (*Candy*)

2 HISTORIQUE

Version	Date	Mise à jour
1.0	01/01/2021	Création du document

3 GÉNÉRALITÉS

3.1TERMINOLOGIE

Le CRTC est un circuit qui passe son temps à compter et à comparer.

Il contient donc principalement des compteurs dont la valeur limite est en général définie dans les registres programmables du circuit.

Dans l'objectif de construire des écrans composés de plusieurs caractères verticaux, eux mêmes constitués plusieurs lignes verticales, elles-mêmes composées de plusieurs caractères horizontaux, certaines terminologies sont associées à ces registres et compteurs.

Il est ainsi question de « *Nombre total horizontal de caractères* », « *Nombre total vertical de caractères* » ou encore « *Raster Maximum* » pour définir la valeur de certains registres, avec parfois quelques différences entre les documentations techniques. Certains émulateurs utilisent des acronymes « non officiels » pour définir le nom des compteurs associés : HCC, VTAC, VLC, VCC,... pour n'en citer que quelques uns.

Ces terminologies ne sont plus du tout adaptées lorsqu'on travaille à un niveau différent de celui prévu pour la construction d'un écran « standard ».

En effet, à un certain niveau, il n'y a plus vraiment de logique de comptage « horizontale » ou « verticale », et il est donc difficile de parler de « caractères » sans se prendre les pieds dans le tapis, car selon la programmation du circuit, un caractère vertical peut aussi être un caractère horizontal.

Cette notion existe tout au plus par rapport à la logique de synchronisation de l'écran construit.

Etant donné qu'il est possible de réduire la taille d'une ligne à 1 µsec, de réduire la taille d'un caractère vertical à 1 ligne et réduire la taille d'un écran à 1 caractère, les appellations horizontales et verticales perdent quelque peu leur sens.

C'est pourquoi dans ce document je vais en général noter les registres du CRTC sous la forme **Rn** et les compteurs associés par **Cn**. Lorsqu'il sera question de « caractères », il s'agira des mots traités par la GATE ARRAY à partir de l'adresse fournie par le CRTC. J'ai nommé VMA et VMA' les deux pointeurs internes du CRTC.

J'invite les auteurs d'émulateurs se basant sur ce document à adopter ces notations.
Voici une équivalence de quelques termes rencontrés :

Noms identifiés dans certains émulateurs	Compteur
HCC (Horizontal Char Counter)	C0
VLC (Vertical Line Counter)	C9
VCC (Vertical Character Counter)	C4
VSC (Vertical Sync Counter)	C3h
HSC (Horizontal Sync Counter)	C3l
VTAC (Vertical Total Adjust Counter)	C5 or C9 (on CRTC 0)
VMA (byte pointer)	VMA or VMA' (word pointer)

3.2ACRONYMES

CRTC : Cathod Ray Tube Controller : Circuit qui permet d'interfacer un ordinateur avec un moniteur cathodique capable de gérer des lignes raster.

VSYNC : Vertical SYNC : Se dit du signal émis par le CRTC qui permet au moniteur (via le GATE ARRAY sur CPC) de synchroniser l'écran verticalement.

VBL : Vertical BLank : Se dit de la période durant laquelle le canon à électron du moniteur, qui a atteint le bas de l'écran, désactive le faisceau pour revenir en haut de l'écran.

HSYNC : Horizontal SYNC : Se dit du signal émis par le CRTC qui permet au moniteur (via le GATE ARRAY sur CPC) de synchroniser l'écran horizontalement.

HBL : Horizontal BLank : Se dit de la période durant laquelle le canon à électron du moniteur, qui a atteint le côté droit du moniteur, désactive le faisceau pour revenir à gauche de l'écran.

VMA : Video Memory Address : Pointeur CRTC sur les mots adressés en mémoire et fourni au GATE ARRAY pour l'affichage.

ASIC : Application Specific Integrated Circuit : Se dit d'un circuit conçu pour une application spécifique. Le GATE ARRAY est un ASIC.

RLAL : Rupture Ligne à Ligne : Se dit d'une rupture horizontale ou l'adresse est changée sur chaque ligne raster de l'écran.

RV : Rupture Verticale : Se dit d'une rupture dont l'axe n'est pas « horizontal », et qui permet de créer des zones horizontalement.

RVI : Rupture Verticale Invisible : Se dit de ruptures verticales ayant lieu durant la Hsync et non visibles.

RVLL : Rupture Verticale Last Line : Se dit de ruptures exploitant le traitement « dernière ligne » de C4 des CRTC 0 et 2.

4 CRTC & CPC...

4.1GÉNÉRALITÉS

Sur AMSTRAD CPC, la durée d'un caractère CRTC est de 1 µsec.

Ce caractère CRTC représente 2 octets en mémoire.

Le pointeur mémoire est communiqué par le CRTC au GATE ARRAY, **qui va toujours lire la "ram centrale" de 64k**. Le GATE ARRAY ne peut pas lire les données en ROM ou dans la RAM additionnelle des 6128 (ou des extensions mémoire), sauf intervention de Tot0.

Le CRTC est programmé de manière à ce que l'image ainsi créée soit supportée par un moniteur.
Un raccourci de langage consiste à utiliser le terme NOP au lieu de µsec (microseconde) car c'est le temps pris par cette instruction en Z80A.

Sur CPC, les instructions Z80A sont alignées en arrondissant les cycles M d'une instruction sur un multiple de 4 cycles T. Cet alignement est lié à la nécessité pour le GATE ARRAY d'interrompre le Z80A pour accéder à la ram dont l'adresse est fournie par le CRTC. Ce fonctionnement ralentit certaines instructions par rapport à la fréquence d'horloge. Réaliser un code précis nécessite de connaitre le temps exact pris par chaque instruction. Voir chapitre 23, page 159, pour le détail de ces durées pour chaque instruction.

Un peu d'histoire

Les moniteurs et téléviseurs cathodiques sont contraints d'aligner leur fréquence d'affichage au réseau de distribution d'électricité du pays de commercialisation.

En Europe notamment, les téléviseurs construits pour supporter les formats SECAM et PAL fonctionnent avec une fréquence horizontale de 15.625Khz, pour une fréquence verticale de 50Hz.

A noter que le format Américain (et Japonais) était le NTSC (National Television System Committee) avec une fréquence horizontale de 17.734Khz et verticale de 60Hz.

La fréquence horizontale de 15.625Khz est issue de l'utilisation de ligne à retard de 64 µs, inventée pour la conception du format Secam par l'ingénieur Henri de France.
Pour rappel, 64µs = 0.000064 seconde (juste le temps de prendre un café serré).

Le CRTC suit cette logique et traite 1 million de caractères par seconde environ car sa fréquence est de 1 Mhz.

Sans intervention diabolique sur la position de la HSYNC, il est en principe programmé pour générer des lignes complètes de 64 µsec.

Pour se rapprocher de la fréquence voulue, le CRTC est en général programmé pour afficher 312 lignes de 64µs, soient très exactement 19968 µsec (0.019968 secondes).

Cette fréquence étant liée à une horloge, on peut constater des dérives entre 2 machines au bout d'un certain temps.

Le format standard initialisé par la ROM CPC dans un pays Européen est :

- Lignes de 64 caractères "CRTC" horizontaux de 1µs chacun (composées de 40 caractères affichés et 24 non affichés).
- 312 lignes verticales, sous divisées en caractères verticaux de 8 lignes, soient 39 caractères.
- Formule : ((R4+1) x (R9+1)) + R5
- Sur ces 39 lignes de caractères, 25 sont affichées.
- Adresse en ROM de la table CRTC : **&5C5**

Le format standard initialisé par la ROM CPC aux Etats-Unis est :

- Lignes de 64 caractères "CRTC" horizontaux de 1µs chacun (composées de 40 caractères affichés et 24 non affichés).
- 262 lignes verticales, sub divisées en caractères verticaux de 8 lignes, soient 32 caractères + 6 lignes d'ajustement
- Formule : ((R4+1) x (R9+1)) + R5
- Sur ces 32.75 lignes de caractères, 25 sont affichées.
- Adresse en ROM de la table CRTC : **&5D5**

Note : C'est le bit 4 du port B du PPI (LK4) qui permet de tester quelle table utiliser.

Note : L'initialisation des registres du CRTC est une des premières choses que la ROM BASSE du CPC effectue à l'allumage et au reset de la machine. L'initialisation des registres commence 64 µsec après la lecture de la première instruction en ROM par le Z80A. Les registres sont initialisés en partant du registre 15 jusqu'au registre 0.

Les moniteurs CTM utilisent les circuits PC1031 (moniteur monochrome vert GT65) et PC1378 (moniteur couleur CTM 644). Ces circuits servent à piloter le déflecteur horizontal pour produire le signal dont le canon a besoin pour le balayage.

Le GATE ARRAY du CPC produit, via son pin 5, un signal de synchronisation composite combinant la VSYNC et la HSYNC, qui arrive au connecteur DIN prévu pour le moniteur (via une résistance 220 Ohm (R137)).

Ce signal composite est décodé par les circuits (PC1031 ou PC1378) du moniteur.
[Le moniteur n'étant pas un téléviseur, les signaux pour l'image sont toujours du RGB]

Le CRTC dispose de registres pour gérer un curseur et lire les données envoyées par un "stylo optique". Les registres relatifs au curseur ne servent pas sur CPC, qui ne gère pas de curseur hardware, en général prévu lorsqu'un mode texte est géré.

Ils peuvent cependant mériter un point d'intérêt, car l'action sur d'autres registres dans ou hors d'une période synchronisation, avec des petites valeurs, pourrait entrainer des conséquences sur d'autres registres.

Le fonctionnement de ces registres n'est pas abordé (pour le moment) dans ce document.

4.2 NUMÉROTATION DES CRTC

AMSTRAD a eu la brillante idée d'utiliser des circuits CRTC 6845 fabriqués par différents constructeurs dans ses machines. Ils ont même conçu des ASIC capables d'émuler son fonctionnement.

Un peu d'histoire

Alors que la société AMSTRAD voulait attaquer le marché américain avec les CPC 6128, un problème a été identifié en ROM avec la valeur du registre 5 (ajustement vertical).

Cette valeur a été fixée à 6 en conformité avec la fréquence souhaitée de 60Hz aux USA (262 lignes), mais faisait (à tort) l'impasse sur le système d'interruption géré par le GATE ARRAY. En effet, sur un CPC Européen, les interruptions débutent 2 lignes (on considèrera qu'on a 1 HSYNC par ligne) après la survenue du signal VSYNC par le CRTC.

Ces interruptions ont une période 52 "lignes", ce qui donne exactement 6 périodes durant les 312 lignes (voir chapitre 25.6 sur les INTERRUPTIONS). Pour caler 5 périodes de 52 lignes, il aurait fallu programmer 260 lignes et non 262 comme cela a été fait, et donc que R5 soit programmé avec 4 au lieu de 6.

Ces 2 lignes de plus provoquent l'arrivée de l'interruption sur la même ligne que celle sur laquelle le CRTC signale le début de la VSYNC, MAIS avant lui.

Autrement dit, sur un CPC américain, un programme dont le code principal teste l'attente VSYNC via le PPI peut être interrompu pendant cette attente. Si l'interruption dure trop longtemps, à son retour, la boucle d'attente de la VSYNC a loupé le signal (le bit est revenu à 0), et un risque de "deadlock" existe.

Cela pouvait engendrer des problèmes de compatibilité pour des programmes produits en Europe. AMSTRAD a alors décidé de remédier au problème en ….modifiant la table 60 Hz en ROM…. Non…je plaisante…trop simple.

Pour éviter ce « deadlock » sans modifier la ROM, les ingénieurs Amstrad se sont dit qu'ils pouvaient "limiter le problème" en augmentant la durée de la VSYNC. C'était parfaitement possible en utilisant le registre 3 du CRTC, qui avait été programmé avec 8 lignes ….. dans la ROM.

Il est probable qu'ils se soient dit qu'il existait des modèles de CRTC dépourvus de la fonction utilisée pour fixer le nombre de lignes de la VSYNC (ces modèles fixent le nombre de lignes à 16) et décidé ainsi que les CPC américains ne seraient équipés que de CRTC 1 et 2, dépourvus de cette fonction.

Si les ingénieurs AMSTRAD étaient conscients de l'existence de ces différences, on peut supposer que les premiers CRTC utilisés étaient des types 0, puisque cette fonction est utilisée et programmée par la ROM.

De là à pouvoir affirmer que sans un bug lié à un 4 à la place d'un 6 dans la ROM, il n'y aurait qu'un seul type de CRTC utilisé dans tous les CPC, c'est faire fi des considérations commerciales sur le marché des composants par rapport au succès de la machine en Europe…

Plusieurs entreprises ont créé des versions différentes du circuit, implémentant des fonctions complémentaires, comme la programmation du nombre de lignes de la VSYNC que je viens d'évoquer.

On peut cependant déduire que les concepteurs de la ROM BASIC :

- travaillaient originellement avec un CRTC disposant de la possibilité de programmer ce nombre de lignes pour la VSYNC.
- ont estimé que 8 lignes de VSYNC étaient suffisantes pour synchroniser verticalement l'image sur un moniteur CTM.

Au-delà des différences fonctionnelles et techniques documentées dans les guides constructeurs des circuits (appelés « datasheet » avec 2 « e »), ces CRTC ont tendance à se comporter différemment lorsqu'on commence à modifier des registres :

- plusieurs fois au cours du balayage.
- durant ou hors des périodes HSYNC/VSYNC.
- avec la valeur 0, qui est un cas particulier pour la gestion de plusieurs registres.

Ces différences impactent la compatibilité des programmes, notamment lorsqu'il est question de modifier l'adresse du pointeur vidéo en cours de balayage.

Cette technique est encore appelée communément "*Rupture*" car… elle est plus simple et générique à dire que "*Offset Split Screen*".

Les différences de gestion des compteurs (qui peuvent déborder dans quelques situations) induisent généralement un défaut de synchronisation horizontale (R2) et/ou verticale (R7).

Si un code Z80A "attend" le signal VSYNC ou qu'il utilise les interruptions (qui dépendent de la HSYNC), le bordel est accentué.

Un peu d'histoire

A ma connaissance, le premier programme a avoir réalisé un test de CRTC est le jeu "Crafton & Xunk" (Get Dexter), écrit par Rémi Herbulot et Michel RHO en 1986.
Dans ce jeu, le changement d'écran a lieu grâce un scrolling horizontal, qui utilise le registre 2 (positionnement de la HSYNC sur un autre caractère).
Cette méthode, sur un CRTC MOTOROLA, provoque une perte de synchro verticale lorsque la VSYNC se produit durant une HSYNC (Ghost Vsync).
Autrement dit, dans cette situation, le CRTC croit faussement générer un signal VSYNC pour le moniteur.
Je suppose que Rémi Herbulot devait avoir accès, chez Ere Informatique, à un CPC avec un CRTC HITACHI (comme le mien à l'époque) et un autre avec un CRTC MOTOROLA.
Ayant fait ce constat, il a créé un test basé sur la lecture en &BF00 du registre 12, qui permet de distinguer le CRTC HITACHI du CRTC MOTOROLA.
Et il a donc géré un affichage avec et sans scrolling de l'écran.
C'est pourquoi il n'y a pas de scrolling également sur les CPC équipés de CRTC UMC, alors que ce CRTC le permet sans problème, car comme pour le CRTC MOTOROLA, son registre 12 n'est pas lisible.
Pour l'anecdote, j'avais retiré le test "anti scroll" sur le CPC d'un ami pour voir ce que ça donnait, et comme ça fonctionnait (il avait sans doute un CRTC 1), je me suis dit que ce test était prévu pour autre chose. J'étais alors loin de me douter que les CRTC étaient différents.

Un peu d'histoire (encore !)

Lorsque les premières techniques de "rupture" ont commencé à être utilisées massivement dans les démos, les différences entre CRTC ont vite commencé à poser problème.
Et notamment lorsque les programmeurs indépendants (souvent lycéens ou étudiants) ont commencé à constituer la scène « démo » et que leurs démos ont commencé à circuler de manière moins discrétionnaire. Au départ, ces premières démos étaient surtout des introductions pour des jeux craqués qui circulaient dans les cours d'école.
[Je crois qu'il existe un jeu des années 80 qui a rencontré ce type de problème avec le CRTC (à vérifier)]

Les premiers programmeurs de démo n'avaient pas encore constitué de "réseau" (la demoscene) et disposaient en général d'une seule machine sous la main.
Les premières intros et démos circulaient au niveau d'un cercle restreint et très régional. Le "réseau" consistait, dans les années 80, à des échanges postaux, minitels (un ancêtre français d'internet, comme prestel en UK) et des échanges téléphoniques ruineux hors des départements, avec très peu ou pas de relations avec les autres pays.

La communication avec des personnes d'autres régions est d'abord passée par la consultation de petites annonces dans les rares magazines informatiques de l'époque à en disposer, car il était difficile de trouver des coordonnées dans une introduction précédant un jeu craqué. Les programmeurs pouvaient difficilement se rendre compte du résultat produit par leur code sur d'autres machines, et l'inertie était énorme.
Il était difficile d'adapter le code via des échanges postaux, sans disposer de la machine (ou même de vouloir le faire, tout simplement).
Les problématiques rencontrées pouvaient aller de la désynchronisation d'image au plantage pur et simple de la machine.

Il était difficile d'être catégorique sur l'origine effective de ces problèmes, et surtout sur leur étendue. Néanmoins, avec le regroupement des demomakers et l'agrandissement de la demoscene, il a été possible de confronter le code sur différentes machines.

La présence de quelques électroniciens dans les rangs des demomakers a permis d'identifier les coupables… (M. SUGAR est toujours en fuite).

Quelques règles "universelles" issues de démarches empiriques ont été décrites dans des documentations secrètes qui ne le sont pas restées longtemps ("Il ne faut pas faire cette opération ici pour que ça fonctionne partout", par exemple).

A noter que certaines techniques sont toujours à l'heure actuelle considérées impossibles à réaliser d'un CRTC à l'autre (jusqu'à ce document…).

Le CRTC 2 (MOTOROLA) s'est vite avéré être celui posant problème pour une des techniques les plus utilisées, qui consiste à placer R4=R9=0.

Le code nécessaire pour assurer la compatibilité entre CRTC peut néanmoins être bien plus complexe que de juste savoir quand modifier un registre 9 ou 4, nous allons le voir…

Afin de pouvoir exploiter les règles permettant la compatibilité, il était nécessaire dans un premier temps de pouvoir identifier les différents CRTC.

De nombreuses méthodes existent aujourd'hui. Voir chapitres 26, page 168, et 27, page 170.

La numérotation est restée à ce jour celle fixée en fonction de l'ordre ou j'ai découvert ces circuits avec l'aide de quelques membres du groupe de demomakers Logon System.

Nous avons découvert le CRTC émulé "Pré ASIC" après la sortie de l'AMSTRAD PLUS, et c'est pourquoi son numéro est supérieur à 3, bien qu'il soit sorti plus tôt chronologiquement.

Type	Marque	Modèle
0	HITACHI	HD6845S(P)
0	UMC	UM6845
1	UMC	UM6845R
2	MOTOROLA	MC6845(P)
3	AMSTRAD	ASIC 40489
4	AMSTRAD	ASIC 40226

Note 1 : Les CRTC 3 et 4 sont des CRTC émulés par des ASIC, mais sont néanmoins des CRTC! Sans activer les fonctions complémentaires de l'ASIC 40489, le comportement de ces deux CRTC ne peut pas être différencié actuellement.

Note 2 : Le "Pré-ASIC" (CRTC 4) avait très certainement été conçu dans l'optique de l'AMSTRAD PLUS, car son compteur C9 est prévu pour être interrompu sur n'importe quelle ligne.
C'est pourquoi positionner R9 à 0 peut se faire sur la dernière ligne d'un caractère (compatibilité CRTC 0) ou sur la première ligne d'un caractère (compatibilité CRTC 1).

Note 3 : Il n'est pas exclu, encore à l'heure actuelle, que certaines séries de CPC, soient équipés de modèles de CRTC différents, disponibles chez les 3 constructeurs. (Merci de me le rapporter si vous découvrez un autre modèle, et que vous êtes sûr que ce n'est pas votre petite sœur qui l'a remplacé pour vous faire une blague).
A priori, le CRTC HD6845S de HITACHI, qui est identifié comme un type 0, se comporte exactement comme le CRTC UM6845 de UMC. Sans doute un échange de bon procédé entre les firmes ou c'est juste le marquage qui a changé...
La documentation UMC le précise dans sa table de comparaison avec les autres circuits.
Il n'y a pas à ce jour de test qui permette la distinction de ces deux circuits, mais c'est peut-être possible via le mode « Interlace » ou la gestion particulière de C0 par le HD6845S.

Note 4 : Programmer R3 avec &8x est une très mauvaise habitude, puisque les CRTC 1 et 2 existent aussi dans les machines Européennes, et qu'ils ne respectent pas cette valeur...

Note 5 : Programmer une modification précise d'un registre CRTC dans une routine d'interruption qui n'interrompt pas un code parfaitement synchronisé est une très mauvaise idée.
(et peut laisser croire (à tort) à une différence de CRTC).

4.3 VUE GENERALE DES REGISTRES

			CRTC 0									CRTC 1, 2										CRTC 3, 4									
Register	**Definition**	**Unit**	**r/w**	7	6	5	4	3	2	1	0	**r/w**	7	6	5	4	3	2	1	0		**r/w**	7	6	5	4	3	2	1	0
R0	Horizontal total character number	Char	w									w										w								
R1	Horizontal displayed character number	Char	w									w										w								
R2	Position of horizontal sync. pulse	Char	w									w										w								
R3	Pulse width of horizontal sync. pulse	Function	w	v	v	v	v	h	h	h	h	w					h	h	h	h		w	v	v	v	v	h	h	h	h
R4	Vertical total character number	Char Row	w									w										w								
R5	Total raster adjust	Scan Line	w									w										w								
R6	Vertical displayed character number	Char Row	w									w										w								
R7	Position of vertical sync. pulse	Char Row	w									w										w								
R8	Interlace Mode and Skew	Function	w	c	c	d	d			i	i	w							i	i		w	c	c	d	d			i	i
R9	Max Scan Line Address	Scan Line	w									w										w								
R10	Cursor start	Scan Line	w		b	p						w		b	p							w		b	p					
R11	Cursor end	Scan Line	w									w										w								
R12	Display start address (High)	Pointer	r/w									w										w								
R13	Display start address (Low)	Pointer	r/w									w										w								
R14	Cursor address (High)	Pointer	r/w									r/w										r/w								
R15	Cursor address (Low)	Pointer	r/w									r/w										r/w								
R16	Light Pen (High)	Pointer	r									r										r								
R17	Light Pen (Low)	Pointer	r									r										r								
La table suivante décrit les ports d'accès au		**CRTC sur C**	**r/w**	7	6	5	4	3	2	1	0	**r/w**	7	6	5	4	3	2	1	0		**r/w**	7	6	5	4	3	2	1	0
&BC00	Sélection N° de registre sur 5 bits	Number	w									w										w								
&BD00	Ecriture donnée du registre	Value	w									w										w								
&BE00	Registre de Status	Function										r		L	b						(*)	r								
&BF00	Registre de Lecture	Value	r									r										r								

(*) CRTC 1 only

4.4 ACCES AU CRTC

L'accès aux entrées/sorties avec un Z80A nécessite en général d'utiliser des instructions spécifiques.

Ces instructions (OUT, OUTI, INI, IND...) sont en principe prévues pour utiliser des périphériques dont les adresses sont définies sur les 8 bits de poids faible du bus d'adresse 16 bits.

Le bus d'adresse 16 bits est spécifié dans le registre **BC**, mais certaines instructions (OUTD, OUTI, INI, IND, ...) utilisent aussi **B** comme compteur.

Dans l'optique de conception d'un ordinateur, il n'est pas conseillé de placer l'adresse des périphériques pouvant utiliser ces instructions sur les 8 bits de poids fort du bus d'adresse.

Ce sage conseil de M. ZILOG n'a pas été écouté par M. SUGAR.

Aussi l'accès aux périphériques sur CPC passe principalement par les bits **A8..A15 du bus d'adresse**. (le FDC 765 faisant partiellement exception)

Les bits de sélection des périphériques doivent en conséquence être positionnés sur B.

Bienvenue sur le CPC!

Exemples		
Selection registre 12		
	BASIC	OUT &BC00,12
	Z80A	LD BC,&BC00+12:OUT(C),C
Envoi valeur &30 sur registre 12 (sélectionné précédemment)		
	BASIC	OUT &BD00,&30
	Z80A	LD BC,&BD30:OUT (C),C
Lecture registre 12 (sélectionné ou mise à jour précédemment)		
	BASIC	PRINT INP(&BF00)
	Z80A	LD BC,&BF00 : IN A,(C)
Lecture registre de status		
	BASIC	PRINT INP(&BE00)
	Z80A	LD BC,&BE00 : IN A,(C)

L'utilisation de B condamne l'utilisation d'instructions intéressantes permettant d'envoyer ou lire des séries de valeurs successives sur un port, comme OTIR, OTDR, INIR, INDR.

En effet ces instructions utilisent B comme compteur du nombre de valeurs à lire dans une table, et décrémentent ce compteur jusqu'à atteindre la valeur 0.

Remarque digressive :

Ces instructions répétitives peuvent être expérimentalement utilisées, pour traiter plus d'une valeur, sur un périphérique dont les bits de sélection ne participent pas au compteur (bits "haut"), comme le GATE ARRAY.
Il est possible de lancer une de ces instructions et de l'interrompre sauvagement en la positionnant judicieusement avant qu'une interruption survienne.
Cette interruption devra néanmoins "oublier" l'adresse de retour placée sur la pile avant de ré-autoriser d'autres interruptions.
L'intérêt est cependant très limité (et surtout ludique) et pourra activer d'autres périphériques selon le nombre de valeurs lues dans la table.

Il reste néanmoins possible d'utiliser

- Les instructions OUTI ou OUTD pour envoyer une par une des données issues d'une table (pointée par HL) au CRTC.
- Les instructions INI ou IND pour lire une par une des données issues d'une table (pointée par HL) du CRTC.

L'intérêt est que ces instructions sont "rapides" au regard du nombre d'opérations effectuées.

L'utilisation de l'instruction OUTI/OUTD est possible en incrémentant préalablement B entre chaque instruction pour OUTI/OUTD car B est décrémenté avant l'accès au port.
Pour le CRTC, qui utilise les bits 0 et 1 de B comme index, cela implique que B soit pré-chargé avec l'adresse du port + 1 pour que les instructions OUTI/OUTD fonctionnent.

Pour les instructions de lecture, le périphérique adressé est défini par BC avant la décrémentation de B, qui doit donc contenir l'adresse normale du port avant lecture.

Exemple en Z80A :

```
                LD BC,&BC02         ; Sélection registre position Hsync
                OUT (C),C
                LD HL,TABHSYNC      ; Pointeur sur la table des positions Hsync
                LD B,&BD+1          ; Port envoi donnée + 1, soit &BE
                OUTI                ; OUTI décrémente B, envoie TABHSYNC[0] sur le port &BD
                                    ; et incrémente le pointeur
                INC B               ; B est remis sur &BE
                OUTI                ; OUTI décrémente B, envoie TABHSYNC[1] sur le port &BD
                                    ; et incrémente le pointeur
TABHSYNC        DB 50, 10
```

Autres instructions en Z80A permettant de traiter une entrée sortie :

INSTRUCTIONS	DUREE	DESCRIPTION
OUT (C),r8	4 µsec	r8=[A ou B ou C ou D ou E ou H ou L] Ecriture sur les périphériques définis dans BC de la valeur contenue dans le registre r8.
OUT (n),A	3 µsec	L'adresse d'entrée sortie est définie par le couple An, et la donnée envoyée au périphérique est A. Cette double contrainte limite drastiquement le nombre de valeurs disponibles pour un périphérique sans provoquer un effet collatéral sur un autre périphérique (sélection+valeur).
OUT (C),0	4 µsec	Ecriture sur les périphériques définis dans BC de la valeur 0. Une instruction intéressante pour un demomaker! Mais pas que...
IN r8,(C)	4 µsec	r8=[A ou B ou C ou D ou E ou H ou L] Lecture dans le registre r8 de la valeur envoyée par le périphérique défini dans BC. Si plusieurs périphériques sont sélectionnés, nul doute qu'un jeu des chaises musicales va avoir lieu...
IN A,(n)	3 µsec	L'adresse d'entrée sortie est définie par le couple An, et la donnée lue va modifier A.
IN (C)	4 µsec	C'est une instruction « non officielle » La valeur présente sur le bus de données est lue et son évaluation affecte F

4.1DELAIS D'ACCES

Le tableau suivant indique pour quelques instructions d'écriture quand le registre est effectivement mis à jour dans le circuit.

INSTRUCTIONS	DUREE	PRISE EN COMPTE I/O
OUT (C),r8	4 µsec	3ème µsec
OUT (n),A	3 µsec	3ème µsec
OUT (C),0	4 µsec	3ème µsec
OUTI	5 µsec	5ème µsec
OUTD	5 µsec	5ème µsec

Il est toutefois important de noter que la prise en compte durant la microseconde de mise à jour n'a pas lieu au même « moment » selon les instructions, et peut donc avoir une incidence sur la gestion de la valeur par le circuit.

Une manière de mesurer cette différence est d'utiliser des processus non « ralentis » par le CRTC et le GATE ARRAY, comme l'affichage de la HSYNC par exemple. Voir chapitre 15, page 92.

On peut aussi mesurer cette différence, par exemple, entre ce qui se produit avec une I/O sur le 3ème NOP d'un OUT(C),R8 et le 5ème NOP d'un OUTI sur un CRTC 1. Voir chapitre 0, page 79.

Il faut aussi noter que les CRTC 3 et 4 retardent la mise à jour des registres de 1 µsec.
Il faut donc que ces instructions surviennent 1 µsec plus tôt que sur un CRTC 0, 1 ou 2.

Par convention dans ce document, j'ai considéré que l'entrée sortie avait lieu 1 µsec plus tard dans l'instruction afin de simplifier les schémas et la compréhension générale.

En pratique, cela impose, pour un code réalisé sur CRTC 3 ou 4, de positionner l'instruction d'entrée-sortie 1 µsec avant celle qu'on aurait placée pour un CRTC 0, 1 ou 2.

Ceci est dit !

5 LES AUTRES CIRCUITS

5.1ACCÈS

Les périphériques, sur CPC, ont été câblés de manière à décoder partiellement l'adresse utilisée pour effectuer l'entrée/sortie. (ai-je déjà écrit « Bienvenue sur CPC » ?).

Un périphérique est concerné par une opération d'entrée/sortie dès que quelques bits précis du bus d'adresse sont positionnés à 0 et/ou 1. Ce qui implique que si d'autres bits relatifs à d'autres périphériques sont également positionnés, alors l'opération d'entrée/sortie va également les concerner.

Il est donc possible d'envoyer une même valeur à différents périphériques **simultanément**.

L'intérêt peut sembler relatif car les valeurs communes utiles à plusieurs périphériques ne sont pas énormes. Cependant, si on mesure bien les conséquences d'envoyer des valeurs imprévues à un périphérique, cela permet de faire des « économies » de registre(s) en modifiant judicieusement la valeur des registres Z80A B et/ou C.

Dans un environnement hautement contraint par le temps machine, cela peut avoir l'intérêt d'affecter d'autres usages à ces registres.

		Registre B (ou A)								Registre C (ou n)								
Circuit	**r/w**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Adr. Sélect Unit.**
PAL	w	**0**	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	&7F00
Gate Array	w	**0**	**1**	x	x	x	x	x	x	x	x	x	x	x	x	x	x	&7F00
CRTC Select	w	x	**0**	x	x	x	x	0	0	x	x	x	x	x	x	x	x	&BC00
CRTC Write	w	x	**0**	x	x	x	x	0	1	x	x	x	x	x	x	x	x	&BD00
CRTC Status	r	x	**0**	x	x	x	x	1	0	x	x	x	x	x	x	x	x	&BE00
CRTC Read	r	x	**0**	x	x	x	x	1	1	x	x	x	x	x	x	x	x	&BF00
ROM Select	w	x	x	**0**	x	x	x	x	x	x	x	x	x	x	x	x	x	&DF00
Printer	w	x	x	x	**0**	x	x	x	x	x	x	x	x	x	x	x	x	&EF00
PPI Port A	r/w	x	x	x	x	**0**	x	0	0	x	x	x	x	x	x	x	x	&F400
PPI Port B	r/w	x	x	x	x	**0**	x	0	1	x	x	x	x	x	x	x	x	&F500
PPI Port C	r/w	x	x	x	x	**0**	x	1	0	x	x	x	x	x	x	x	x	&F600
PPI Control	w	x	x	x	x	**0**	x	1	1	x	x	x	x	x	x	x	x	&F700
FDC Status	r	x	x	x	x	x	**0**	x	1	0	x	x	x	x	x	x	0	&FB7E
FDC Data	r/w	x	x	x	x	x	**0**	x	1	0	x	x	x	x	x	x	1	&FB7F
FDC Motor	w	x	x	x	x	x	**0**	x	0	0	x	x	x	x	x	x	x	&FA7F

Enfin, toujours dans la perspective d'environnements contraints, il peut être utile de considérer que tous les bits d'une valeur envoyée à un périphérique ne sont pas utiles.

Pour un CRTC, c'est vrai pour une donnée de mise à jour d'un registre, comme R9 tronqué à 5 bits, tout comme la valeur du numéro de registre.

Autrement dit, on peut sélectionner R9 avec la valeur 9, mais aussi &31 (00110001), et le mettre à jour avec la valeur 7, mais aussi &27 (00100111).

5.2 CPC + / GX 4000

Sur l'AMSTRAD PLUS, il est possible de communiquer directement avec le CRTC émulé, via des registres complémentaires créés spécifiquement pour gérer les fonctions complémentaires "PLUS". Ces fonctions permettent par exemple de faire des masquages de données, d'effectuer des décalages fins, de définir des lignes et adresses de rupture, ou même de caler une interruption raster ou "dma" (non sans bug toutefois).

L'objectif actuel de ce document n'est pas (encore) d'analyser l'interaction entre ces registres et l'émulation faite du CRTC. L'accès à ces registres spécifiques ne passe pas par le système d'entrée/sortie du Z80A décrit précédemment.

L'AMSTRAD PLUS dispose d'une page de registres dit "mappés" sur une adresse.
Chaque registre dispose donc de sa propre adresse.
Le Z80A accède à ces registres avec de simples écritures (ou lecture) à une adresse donnée.

Il est donc possible d'accéder à certaines fonctions (par exemple la modification d'une couleur) de deux manières différentes. Les informations indiquées pour le CRTC 3 dans cette version du document concernent uniquement des I/O réalisées via les instructions OUT du Z80A.

A noter cependant que la page de ces registres, située entre &4000 et &7FFF, est accessible via une fonction jusqu'alors inexploitée du GATE ARRAY.

Cette fonction est cependant elle-même conditionnée à l'utilisation d'une séquence de « déverrouillage » de 17 octets qui doit être envoyée sur le port de sélection (&BC00) du CRTC.

255, 0, 255, 119, 179, 81, 168, 212, 98, 57, 156, 70, 43, 21, 138, 205, 238

Remarque :
Il ne sert à rien d'envoyer cette séquence sur un CRTC 4.
Ils n'ont pas utilisé la même séquence.
Je ne dirais rien de plus sans la présence de mon avocat...

6 CONSTRUCTION D'UN ÉCRAN

6.1LOGIQUE GENERALE

Les algorithmes suivants décrivent la logique générale de gestion d'affichage avec un CRTC « idéal ». Nous le verrons plus loin, cette logique est parfois très différente selon les situations.

6.1.1 COMPTAGE DES CARACTERES

C0 est incrémenté
Lorsque C0 atteint R0
    C0 passe à 0
    C9 est incrémenté
    Lorsque C9 atteint R9
        C9 passe à 0
        C4 est incrémenté
        Lorsque C4 atteint R4
            Lorsque C5 atteint R5
                C4 passe à 0, C5 passe à 0
                MA est rechargé à partir de R12/R13
            Sinon C5 est incrémenté

6.1.2 SYNCHRONISATIONS

Lorsque C0 atteint R2
    Hsync débute, C3h=0
    C3h est incrémenté si R3h>0
    Lorsque C3h=R3h
        Fin de Hsync

Lorsque C4 atteint R7
    Vsync débute, C3v=0
    C3v est incrémenté
    Lorsque C3v atteint R3v (ou 16)
        Fin de Vsync

6.1.3 AFFICHAGE DES CARACTERES

Lorsque C0 passe à 0
    L'affichage des caractères est activé
Lorsque C0 atteint R1
    L'affichage des caractères est désactivé

Lorsque C4 passe à 0
    L'affichage des lignes de caractères est activé
Lorsque C4 atteint R6
    L'affichage des lignes de caractères est désactivé

6.1.4 POINTEUR VIDEO

A chaque µsec, VMA est incrémenté de 2 tant que l'affichage est actif et C9 intervient dans l'adresse

Si C0=R0 alors C0=0 vrai pour R0=0
Si C9=R9 Alors C9=0 vrai pour R9=0
Si C4=R4 alors
Si R5=0 alors vrai pour R5=0
C4=0 vrai pour R4=0
MA=R12/R13
Sinon C4=C4+1 (crtc 0, 1, 2) et Gestion C5

Sinon C4=C4+1
Sinon C9=C9+1
sinon C0=C0+1

6.1.5 REPRÉSENTATION SCHÉMATIQUE

Les schémas suivants décrivent globalement la construction d'un écran à partir des différents registres, sans que les registres soient modifiés durant le balayage.

Définition Ecran :	Horiz. Reg				Vertical Reg					Vid Ptr		Special
	R0	R1	R2	R3	R4	R5	R6	R7	R9	R12/R13		R8
	63	40	46	14	35	24	25	30	7	&30	0	0

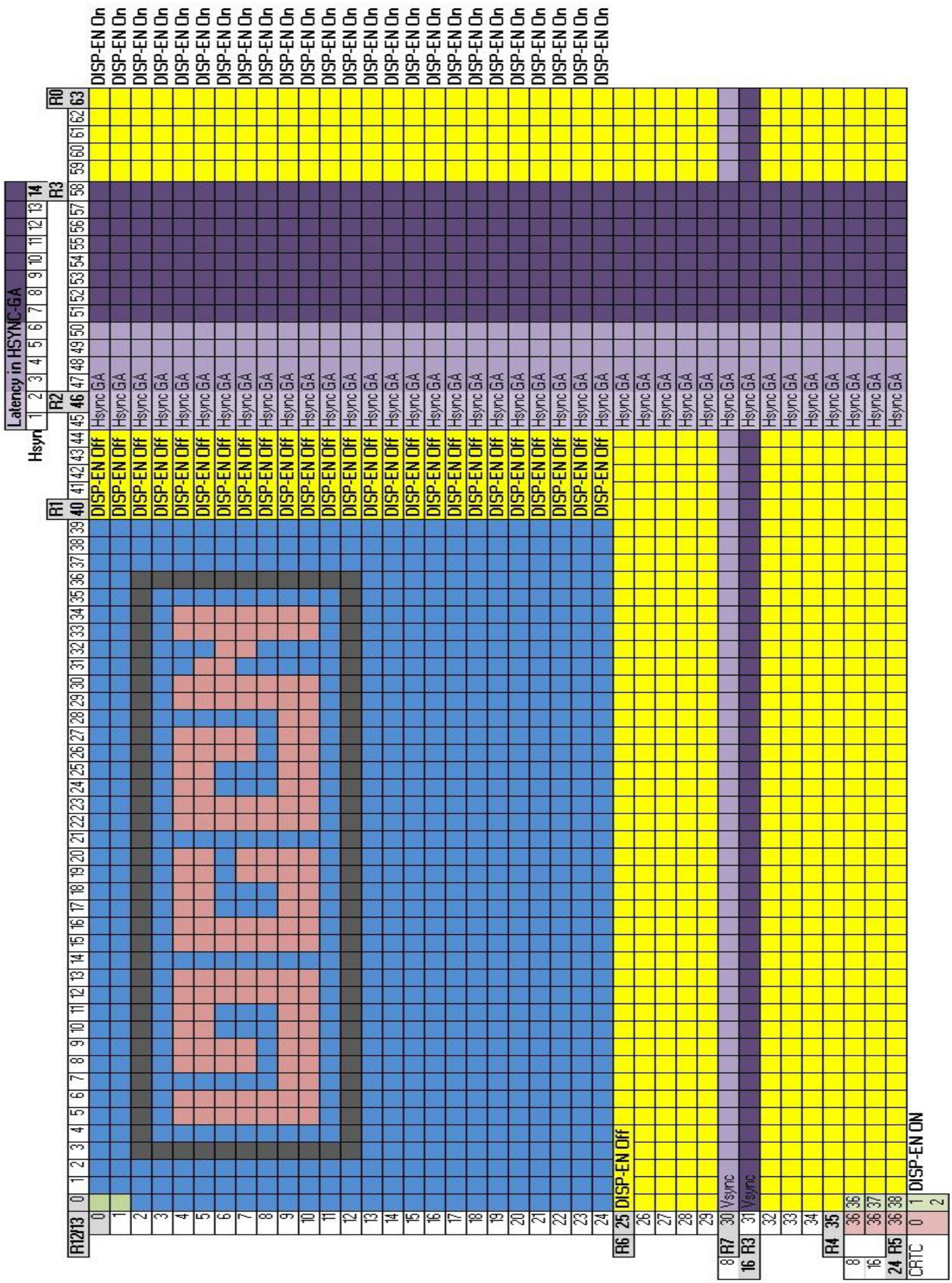
(R0+1) x [((R4+1) x (R9+1)) + R5]
64 x ((36 x 8)+24) = 19968 µs

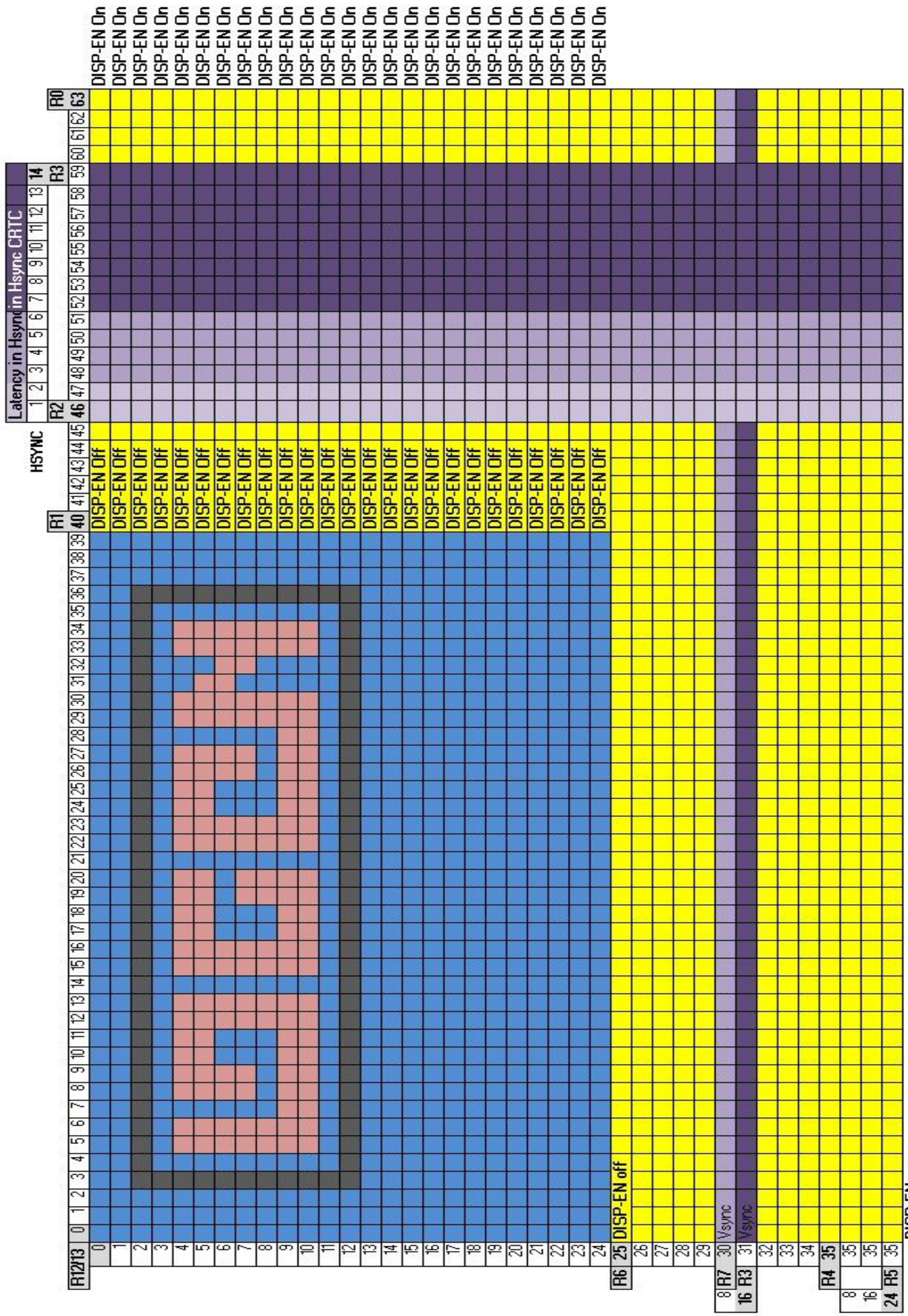
Description des 16 premières lignes en vert indiquées sur les schémas pages suivantes :

Video Pointer:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																	
R12		C9 (*)			R12/13											0	Hex		First byte							Second byte							
5	4	2	1	0	9	8	7	6	5	4	3	2	1	0	0	Addr	C9	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	C4
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C000	0																	0
1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	C800	1																	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	D000	2																	0
1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	D800	3																	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	E000	4																	0
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	E800	5																	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	F000	6																	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	F800	R9 7																	0
1	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	C050	0																	1
1	1	0	0	1	0	0	0	0	1	0	1	0	0	0	0	C850	1																	1
1	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	D050	2																	1
1	1	0	1	1	0	0	0	0	1	0	1	0	0	0	0	D850	3																	1
1	1	1	0	0	0	0	0	0	1	0	1	0	0	0	0	E050	4																	1
1	1	1	0	1	0	0	0	0	1	0	1	0	0	0	0	E850	5																	1
1	1	1	1	0	0	0	0	0	1	0	1	0	0	0	0	F050	6																	1
1	1	1	1	1	0	0	0	0	1	0	1	0	0	0	0	F850	R9 7																	1

(*) C5 on CRTC 3, 4 in vertical adjustment period (R5)


Latency in HSYNC-GA
Hsyn
R0
R1
R2
R3
R12/13
R6
R7
R4
R5
DISP-EN On
DISP-EN Off
Hsync GA
Vsync
CRTC
DISP-EN ON


Latency in Hsync CRTC
HSYNC
DISP-EN On
DISP-EN Off
Vsync
DISP-EN off
DISP-EN on

7 SYNCHRONISATION

7.1PRINCIPES

Le CRTC est chargé de déterminer les adresses mémoire à afficher, qu'il communique au GATE ARRAY, qui lit les données et les gère selon ses paramètres, pour générer des pixels plus ou moins larges et colorés.

Le CRTC envoie aussi au GATE ARRAY les signaux VSYNC et HSYNC, que ce dernier recombine pour envoyer un signal composite au moniteur.

- Lorsque le GATE ARRAY reçoit un signal HSYNC, il y a 2 µs de latence avant que la "vraie" synchro horizontale du moniteur débute.
- Lorsque le GATE ARRAY reçoit un signal VSYNC, il y a 2 lignes de latence avant que la "vraie" synchro verticale du moniteur débute.

Sur un sujet qui traite des différences entre des modèles de circuits à l'échelle de la microseconde, il est important de spatialiser les instructions par rapport à ce qui est affiché.

Plusieurs difficultés sont liées à cette démarche :

- Situer une instruction Z80A par rapport au fonctionnement interne du CRTC, dans la mesure où des délais peuvent exister entre la mise à jour des registres internes et le moment où les caractères sont affichés.
- Déterminer quand, durant l'instruction d'entrée sortie, la modification d'un registre CRTC est effective et prise en compte.

Le même type de question existe pour les instructions Z80A accédant directement au GATE ARRAY, ou lors de la mise à jour de la RAM lue par le GATE ARRAY.

Pour « spatialiser » les instructions, ce document se base sur un point de référence unique sur lequel sont alignées les instructions en Z80A affectant la vidéo.

Ce point de référence est le moment ou on considère que C0=0 au sens CRTC.

Il existe un différé d'affichage entre le moment ou le CRTC fournit un pointeur vidéo, et le moment ou le GATE ARRAY affiche le caractère 16 bits correspondant.
Ce différé est de 1 µsec.

Ce décalage temporel d'affichage du GATE ARRAY par rapport au CRTC ne serait pas gênant si l'intégralité de ce qui est envoyé par le CRTC au GATE ARRAY était **toujours** retardé de 1 µsec.

Mais ce n'est pas toujours le cas, en particulier pour la gestion du signal HSYNC pour les machines équipées des CRTC 0, 1 et 2.

Lorsque ce document traite d'instructions Z80A, et que leur spatialisation a une importance au regard des différences de délai, le compteur C0 est présenté sur 2 "time-line".
Celle au regard de C0 à partir du point de référence CRTC et celle au regard de l'affichage (pixels, ...) par le GATE ARRAY.

Il sera donc question de "**C0 from Vsync**" (ou **C0vs**) et/ou "C0 from GA" lorsque l'affectation des registres du CRTC est déterminante par rapport à l'affichage des caractères par le GATE ARRAY.

Le CRTC communique avec le Z80A de deux manières différentes (en dehors des registres en lecture ou du registre de statut du CRTC 1) :

- Soit via la broche VSYNC du CRTC qui est reliée directement au bit 0 du port B du PPI 8255.
- Soit via les interruptions du Z80A produites par le GATE ARRAY à partir des signaux HSYNC du CRTC.

7.2 SYNCHRONISATION VSYNC

Le CRTC génère un signal sur sa broche VSYNC une µsec avant que C4=R7 et que C0 passe à 0.
C'est un indicateur fiable pour baser ensuite toute autre méthode de synchronisation.

Le bit 0 du port B du PPI (adresse d'entrée sortie "standard" &F5) passe à 1 dès que le signal VSYNC est produit par le CRTC.

En général, les programmes font des "boucles" pour attendre que ce bit passe à 1, ce qui entraine une marge d'erreur liée à la durée des instructions du code d'attente.

Cette marge existe même si le code de « boucles » est "aligné" grâce à une interruption générée à partir d'une instruction HALT située avant la boucle d'attente (la marge est juste "stable").

Il est cependant parfaitement possible d'écrire un code capable de caler du code Z80A sur la microseconde qui correspond à C4=R7 et C0=C9=0.

Toutes les données indiquées dans ce document utiliseront ce point de référence **C0vs**.
Il est bien sûr possible d'utiliser le système d'interruption comme nouveau point de référence.
Le chapitre 25.6.6 sur les interruptions fait une synthèse selon les différents CRTC.

Il ne faut en effet pas perdre de vue que les interruptions sont dépendantes des différences avec les couples CRTC/GATE ARRAY des différentes machines.

Le principe du code de synchronisation **C0vs** est de placer une attente de VSYNC dans une période ou l'indicateur au niveau du PPI n'est pas encore positionné à 1.

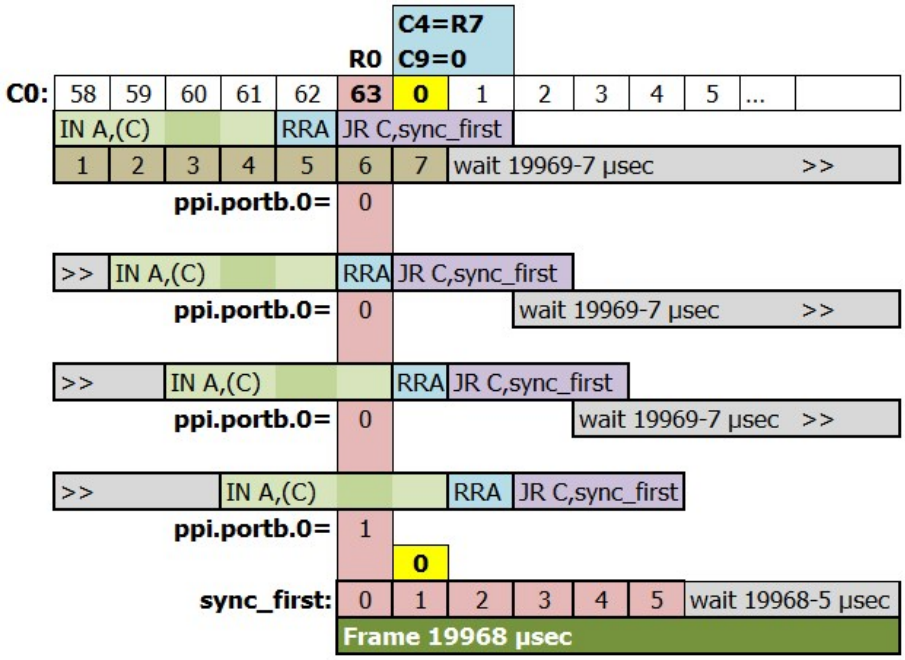
Il s'agit ensuite de faire dériver ce code microseconde par microseconde à chaque frame, grâce à une période de **19969** microsecondes. Ce balayage est plus long de 1 microseconde que celui produit par le CRTC (avec une programmation adéquate et standard de 312 lignes de 64µsec).

Ceci peut être vu comme une "dérive" du code microseconde par microseconde.

L'instruction d'entrée sortie, qui lit le port B du PPI, est l'instruction IN A,(C) et s'exécute en 4 µsec.

J'ai considéré que c'est sur la 3ème µsec de l'instruction IN A,(C) que A est mis à jour lorsque le bit 0 du port B passe à 1. Sans présumer du moment où A est affecté réellement dans l'instruction IN, qui peut aussi dépendre du PPI 8255, il faut juste considérer que le point de référence **C0vs** est calculé de cette manière, et que toutes les informations indiquées dans ce document sont corrélées à cette mesure, qui est un point de référence unique.

Description schématique du principe :



Code de synchronisation :

```
;=============================================================================================
;
;       Synchronisation Vsync
;       A l'issue du call de synchronisation, la 1ere usec est celle sur laquelle le CRTC a positionné le signal Vsync
;       Principe
;               - attendre le signal vsync avec la marge d'erreur
;               - attendre que le signal vsync se termine
;               - faire deriver le test de vsync avec une boucle durant 19968+1 us
;               - des que la vsync est detectee par le test, c'est la detection au plus tot, a laquelle il faut soustraire la duree du test
;
;=============================================================================================
sync_vbl
                di
                ld b,#f5            ; Attendre flag VBL =1
                ld hl,19968-23      ; Compteur de nop (moins les marges et la gestion de l'attente)
                ld de,-11
sync_wvblon1                        ; Ici on attend le debut de la periode VBL (ou on attend pas si on y etait deja)
                in a,(c)            ;
                rra                 ;
                jr nc,sync_wvblon1
sync_wvbloff1                       ; Flag Vsync CRT passe a 1 (ou etait deja a 1)
                in a,(c)            ; Attendre que le flag repasse a 0 (Fin de Vsync)
                rra
                jr c,sync_wvbloff1
sync_wvblon2                        ; On est certain maintenant que le signal Vsync n'etait pas deja en cours
                in a,(c)            ;
                rra                 ; marge1 de 7us
                jr nc,sync_wvblon2
sync_wvbloff2                       ; Attendre que le signal Vsync repasse a 0 en comptant le temps ecoule
                add hl,de           ; 3     On nop2         On nop3
                in a,(c)            ; 4     2               1
                rra                 ; 1     1               1
                jr c,sync_wvbloff2  ; 3/2   2               3       (bcl)+3+4+1+2=15 / marge 15-5=10
                ex de,hl            ; 1
                call wait_usec      ; 5 >> 6 + 10(marge2)
                ;
                ; Zone de  derive pour attendre de nouveau la premiere manifestation du flag
                ; le in a,(c) va "descendre" nop par nop (frame par frame) jusqu'a ce que le in recupere le flag actif
sync_derive_bcl
                ld b,#f5                ; 2
                in a,(c)                ; 4 usec. 0.1.2.[3] (+1)
                rra                     ; 1 usec (+1)
                jr c,sync_first         ; 2/3 (+3)
                ld de,19969-20          ; 3
```

```
                call wait_usec          ; 5+(19969-20)
                jr sync_derive_bcl      ; 3 >> 20
sync_first                              ; 6 Le flag a ete détecté au plus tôt, et ce depuis 5 usec (2+1+3) C0=R0
                                        ; -1 µs pour se positionner sur C0=0
                ld de,19968-11          ; 3
                jp wait_usec            ; 3 >> 11 >> de=19968-11



;================================================================================================
; wait "de" usec
; 40+(((de/8)-5) x 8)+(de and 7) nop
; nb - le call de la fonction n'est pas compte
;================================================================================================
wait_usec:
                ld hl,sync_adjust       ; 3
                ld b,0                  ; 2
                ld a,e                  ; 1
                and %111                        ; 2>8
                ld c,a                  ; 1
                sbc hl,bc               ; 4
                srl d                   ; 2
                rr e                    ; 2>17
                srl d                   ; 2
                rr e                    ; 2
                srl d                   ; 2
                rr e                    ; 2>25
                dec de                  ; 2>27 8
                dec de                  ; 2>29 16
                dec de                  ; 2>31 24
                dec de                  ; 2>33 32
                dec de                  ; 2>35 40 *
                nop                     ; 1>36
wait_usec_01
                dec de                  ; 2 -
                ld a,d                  ; 1 -
                or e                    ; 1 -
                nop                     ; 1 -
                jp nz,wait_usec_01      ; 3 - v=(8 x DE)
                jp (hl)                 ; 1>37
                nop                     ; 1 * v=0--7
                nop                     ; 1
                nop                     ; 1
                nop                     ; 1
                nop                     ; 1
                nop                     ; 1
                nop                     ; 1
sync_adjust
                ret                     ; 3>40 *
```

7.3 FAKE VSYNC

Le signal VSYNC est au sein d'une relation tripartite, entre le CRTC, le GATE ARRAY et le PPI.
Le CRTC communique l'état de sa broche VSYNC au PPI et au GATE ARRAY.

Le PPI 8255 est un circuit générique programmable d'interfaçage de périphériques.

AMSTRAD a utilisé 3 versions différentes de ce composant :

- NEC D8255AC-5
- NEC D8255AC-2
- TOSHIBA TMP8255AP-5

Le dernier chiffre indiqué est la fréquence maximum du circuit.
A ma connaissance il n'existe pas de test pertinent pour différencier ces modèles.

Il est possible de faire fonctionner les broches de communication de ce circuit en entrée ou en sortie selon une programmation adéquate.

Le port B du PPI est conçu sur CPC pour fonctionner en entrée, et donc recevoir des informations des périphériques auquel il est connecté. Il est possible de reprogrammer ce port en sortie, en mettant à 0 le bit 1 du registre de contrôle du PPI situé à l'adresse d'I/O #F700. Une écriture sur le port B (#F500) d'une valeur avec le bit 0=1 va donc mettre la broche 25 du circuit à l'état haut. Cela signifie que le PPI va envoyer son signal vers le CRTC, qui par voie de fait, va le renvoyer vers le GATE ARRAY, puisqu'il y est également connecté.

Cependant, j'ai constaté, comme ArnoldEmu avant moi, que cette FAKE VSYNC ne fonctionne pas sur certains CPC. Sur d'autres CPC, le résultat est incorrect.

Sur un CRTC 2, programmé avec R2=50 et R3=14, la VSYNC n'est en principe plus générée.

Nous verrons plus loin qu'un des problèmes du CRTC 2 est qu'une VSYNC FANTOME est générée si la condition VSYNC a lieu durant la HSYNC. Il devrait donc en théorie être possible de faire ce travail à la place du CRTC en plaçant le bit 0 du port B à 1 au bon endroit et pendant la bonne durée.

Cependant, si on active pendant 1024 µsec une VSYNC PPI en lieu et place de celle qu'aurait généré le CRTC, le signal produit par le PPI n'est à priori pas assez fort pour contrer le signal bas généré par une VSYNC FANTÔME. Mais lorsque le CRTC refait passer sa broche à l'état bas (alors qu'elle y est déjà), alors la VSYNC du PPI est alors prise en compte, et l'image se retrouve stabilisée 16 lignes plus haut que prévu. Un électronicien m'aiderait certainement à y voir plus clair.

Il reste néanmoins possible de contourner plus simplement la VSYNC FANTOME du CRTC 2, en évitant que la condition de démarrage ait lieu durant la HSYNC et en mettant à jour R7 avec C4 juste après la fin de la HSYNC (voir chapitre 15.2, page 92).

ATTENTION : Je ne connais pas vraiment les conséquences « techniques » potentielles relatives à l'envoi d'un signal dans le sens opposé à celui prévu. Ceci étant dit, le risque parait assez faible dans la mesure où, depuis plusieurs années, de nombreux CPC contiennent plusieurs CRTC soudés

l'un sur l'autre dans une orgie d'étain. Le ou les CRTC passifs prenant alors les signaux du CRTC actif. Dans le pire des cas, un CRTC vaut (encore) moins de 10€.

Un peu d'histoire

Ce n'est pas la première fois qu'un port prévu en entrée est utilisé en sortie sur le CPC. Les plus anciens se souviendront sans doute du kit de téléchargement qui permettait de récupérer des programmes (payants) via une connexion téléphonique reliée au minitel (je ne sais pas si ce système a été utilisé avec Prestel en U.K.). Le câble d'un de ces kits était connecté à partir de la prise minitel sur le CPC avec sa prise JOYSTICK, et afin de pouvoir gérer un échange bidirectionnel, le port clavier était alors placé en sortie. Les lignes claviers sont lues à travers le port A du PPI, via le port A du AY-3-8912 (générateur sonore). Même si pour la très grande majorité des utilisateurs, cela n'a eu aucune conséquence, j'ai été témoin de plusieurs CPC dont le AY-3-8912 avait « perdu définitivement » une partie des bits de certains registres du AY. De mémoire, il me semble que ces CPC avait des tonalités particulièrement aigues, si il s'agit de déterminer quels bits étaient « ravagés ».

8 AFFICHAGE, Z80A & GATE ARRAY

Les schémas suivants décrivent, par rapport à quelques instructions de mise à jour de la ram vidéo par le Z80A, quand cette mise à jour est prise en compte par le GATE ARRAY/ASIC.

Les instructions sont localisées par rapport au point de référence C0vs du CRTC.

La lecture en ram par le GATE ARRAY/ASIC pour les données modifiées par le Z80A est identique pour tous les CPC.

8.1 INSTRUCTION LD(HL),reg8 (2 µsec)

HL=0000	reg8=#FF																			
C0vs	00		01		02		03		04		05		06		07		08		09	
Video Ptr	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Z80A Inst	LD (HL),reg8				NOP		NOP		NOP		NOP		NOP		NOP		**LD (HL),#00**			19968-10 µsec
Displayed :	**00**																			

HL=0001	reg8=#FF																			
C0vs	00		01		02		03		04		05		06		07		08		09	
Video Ptr	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Z80A Inst	LD (HL),reg8				NOP		NOP		NOP		NOP		NOP		NOP		**LD (HL),#00**			19968-10 µsec
Displayed :			**00**																	

HL=0002	**reg8=#FF**																			
C0vs	00		01		02		03		04		05		06		07		08		09	
Video Ptr	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Z80A Inst	**LD (HL),reg8**				NOP		NOP		NOP		NOP		NOP		NOP		LD (HL),#00			19968-10 µsec
Displayed :			**FF**																	

HL=0003	**reg8=#FF**																			
C0vs	00		01		02		03		04		05		06		07		08		09	
Video Ptr	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Z80A Inst	**LD (HL),reg8**				NOP		NOP		NOP		NOP		NOP		NOP		LD (HL),#00			19968-10 µsec
Displayed :				**FF**																

8.2 INSTRUCTION LD (aaaa),HL (5 µsec)

aaaa=0004	H=#FF, L=#55																																**H'=0, L'=0**										
C0vs	00		01		02		03		04		05		06		07		08		09		10		11		12		13		14		15		16										
Video Ptr	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33										
Z80A Inst	LD (aaaa),HL										NOP		NOP		NOP		NOP		NOP		NOP		EXX		**Ld (aaaa),HL**																	19968-17 µsec
Displayed :					**00**	**00**																																				

aaaa=0005	H=#FF, L=#55																																**H'=0, L'=0**										
C0vs	00		01		02		03		04		05		06		07		08		09		10		11		12		13		14		15		16										
Video Ptr	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33										
Z80A Inst	LD (aaaa),HL										NOP		NOP		NOP		NOP		NOP		NOP		EXX		**Ld (aaaa),HL**																	19968-17 µsec
Displayed :							**00**	**00**																																		

aaaa=0006	H=#FF, **L=#55**																																**H'=0**, L'=0										
C0vs	00		01		02		03		04		05		06		07		08		09		10		11		12		13		14		15		16										
Video Ptr	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33										
Z80A Inst	**LD (aaaa),HL**										NOP		NOP		NOP		NOP		NOP		NOP		EXX		**Ld (aaaa),HL**																	19968-17 µsec
Displayed :							**55**	**00**																																		
							L																																			

aaaa=0007	H=#FF, L=#55																																H'=0, L'=0										
C0vs	00		01		02		03		04		05		06		07		08		09		10		11		12		13		14		15		16										
Video Ptr	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33										
Z80A Inst	**LD (aaaa),HL**										NOP		NOP		NOP		NOP		NOP		NOP		EXX		Ld (aaaa),HL																	19968-17 µsec
Displayed :									**55**	**FF**																																
									L	**H**																																

8.3 INSTRUCTION PUSH reg16 (4 µsec)

SP=0004	D=#FF, E=#55																																**H=0, L=0, aaaa=0003**					
C0vs	00				01				02				03				04				05				06				07				08				09			
Video Ptr	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33						
Z80A Inst	PUSH DE																NOP				NOP				NOP				NOP				NOP				NOP			
Displayed :					**00**	**00**																																		

19968-17 µsec

SP=0005	D=#FF, E=#55																																**H=0, L=0, aaaa=0004**					
C0vs	00				01				02				03				04				05				06				07				08				09			
Video Ptr	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33						
Z80A Inst	PUSH DE																NOP				NOP				NOP				NOP				NOP				NOP			
Displayed :							**00**	**FF**																																	
								D																																	

19968-17 µsec

SP=0006	D=#FF, E=#55																																**H=0, L=0, aaaa=0005**					
C0vs	00				01				02				03				04				05				06				07				08				09			
Video Ptr	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33						
Z80A Inst	PUSH DE																NOP				NOP				NOP				NOP				NOP				NOP			
Displayed :									**00**	**FF**																															
										D																															

19968-17 µsec

SP=0007	D=#FF, E=#55																																**H=0, L=0, aaaa=0006**					
C0vs	00				01				02				03				04				05				06				07				08				09			
Video Ptr	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33						
Z80A Inst	PUSH DE																NOP				NOP				NOP				NOP				NOP				NOP			
Displayed :											**00**	**FF**																													
												D																													

19968-17 µsec

SP=0008	D=#FF, E=#55																																**H=0, L=0, aaaa=0007**					
C0vs	00				01				02				03				04				05				06				07				08				09			
Video Ptr	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33						
Z80A Inst	PUSH DE																NOP				NOP				NOP				NOP				NOP				NOP			
Displayed :													**55**	**FF**																											
													E	**D**																											

19968-17 µsec

9 GATE ARRAY

Sur CPC, il existe 5 modèles de GATE ARRAY identifiés, chargés, entre autres choses, d'afficher les pixels selon le mode graphique et la couleur définie pour chaque encre.

- Les modèles 40007 et 40008 ont été principalement utilisés sur les CPC 464 et 664 .
- Le modèle 40010 équipe à priori tous les CPC 6128.
- L'asic 40226 est utilisé sur les CPC Low Cost (CRTC 4).
- L'asic 40489 (CRTC 3) est utilisé sur les CPC+ et la GX 4000.

9.1 PIXELISATION

Le GATE ARRAY/ASIC est chargé de convertir la mémoire lue par le CRTC en pixels.

Il effectue cette opération en tenant compte de deux paramètres : le mode graphique et la couleur associée au pixel à afficher.

Ce document n'est pas consacré au GATE ARRAY en détail, mais il faut retenir que le CPC dispose de 4 modes graphiques, dont 3 « officiels ».

Le mode graphique détermine la taille horizontale d'un pixel (en fonction du nombre de couleurs que ce pixel peut prendre).

Un pixel peut être codé sur 1, 2 ou 4 bits.
Un pixel occupant 1 bit est la résolution la plus fine que les GATE ARRAY peuvent produire.

	VRAM BYTE								Displayed Pixels								Definition
Mode	7	6	5	4	3	2	1	0									
0	A0	B0	A2	B2	A1	B1	A3	B3	A3	A2	A1	A0	B3	B2	B1	B0	2 pixels (16 colors)
1	A0	B0	C0	D0	A1	B1	C1	D1	A1	A0	B1	B0	C1	C0	D1	D0	4 pixels (4 colors)
2	A0	B0	C0	D0	E0	F0	G0	H0	A0	B0	C0	D0	E0	F0	G0	H0	8 pixels (2 colors)
3	A0	B0	x	x	A1	B1	x	x	0	0	A1	A0	0	0	B1	B0	2 pixels (4 colors)

A 17th color, stored in the GATE ARRAY, is displayed when BORDER is activated

Cette codification du pixel représente un index dans une table contenant des couleurs codées sur 5 bits.

				INKR			Color				
Color Index				7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	C1	C1	C1	C1	C1
0	0	0	1	0	1	0	C2	C2	C2	C2	C2
0	0	1	0	0	1	0	C3	C3	C3	C3	C3
0	0	1	1	0	1	0	C4	C4	C4	C4	C4
0	1	0	0	0	1	0	C5	C5	C5	C5	C5
0	1	0	1	0	1	0	C6	C6	C6	C6	C6
0	1	1	0	0	1	0	C7	C7	C7	C7	C7
0	1	1	1	0	1	0	C8	C8	C8	C8	C8
1	0	0	0	0	1	0	C9	C9	C9	C9	C9
1	0	0	1	0	1	0	C10	C10	C10	C10	C10
1	0	1	0	0	1	0	C11	C11	C11	C11	C11
1	0	1	1	0	1	0	C12	C12	C12	C12	C12
1	1	0	0	0	1	0	C13	C13	C13	C13	C13
1	1	0	1	0	1	0	C14	C14	C14	C14	C14
1	1	1	0	0	1	0	C15	C15	C15	C15	C15
1	1	1	1	0	1	0	C16	C16	C16	C16	C16

Lorsque le GATE ARRAY affiche ses pixels, il est trop "rapide" pour ceux du MODE 2.

En effet, sur les GATE ARRAY 40007, 40008, 40010 et ASIC 40226 (CRTC 4), les pixels en mode 2 sont affichés un pixel plus tôt que pour les autres modes graphiques.

Ils sont affichés en "avance" de 1/16 µsec (0.0625 µsec).

Autrement dit, le BORDER "cesse" 1 pixel plus tôt sur une ligne en mode 2, et débute 1 pixel plus tôt lorsque C0=R1 (si on n’a pas changé de mode graphique en cours de ligne).

Etant donné qu'il est possible de changer de mode graphique entre chaque ligne et/ou durant une ligne, il faut en tenir compte s’il est nécessaire d'aligner des pixels affichés dans des modes graphiques différents.

L'ASIC 40489 du CPC+ n'est pas concerné par ce décalage.
Quelque soit le mode graphique sur cette machine, les pixels sont alignés.

9.2INKERISATION

Lorsque la table des couleurs (couleur d'une encre et couleur du border) est mise à jour, la prise en compte du changement de couleur n'est pas exactement identique selon les CRTC.

Cependant, le moment ou la mise à jour a lieu est identique quelque soit le mode.

9.3 VITESSE DE PRISE EN COMPTE

Les deux schémas pages suivantes décrivent la prise en compte de différentes instructions mettant à jour la couleur d'une encre et sa prise en compte par le GATE ARRAY/ASIC lorsqu'il affiche ses pixels.

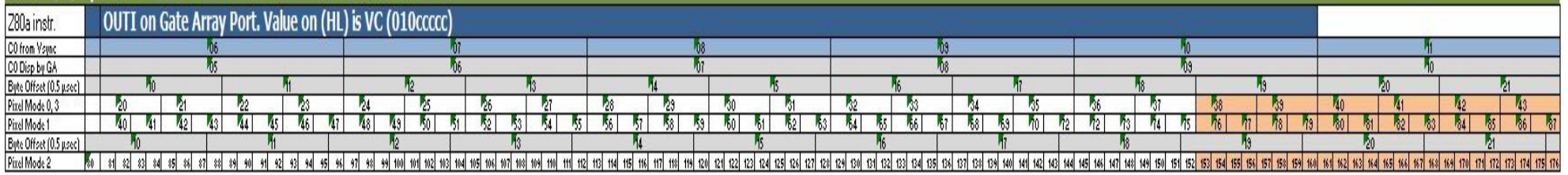
Cette couleur sur les schémas indique quels pixels selon le mode sont affectés par le changement de couleur.

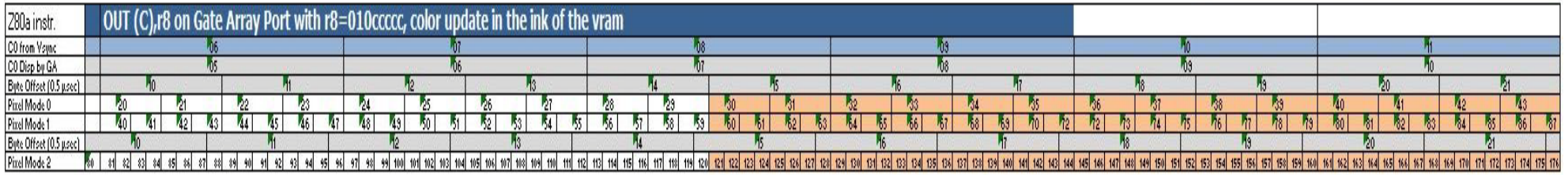
Les instructions sont calées selon **C0vs** calculé à partir du point de référence VSYNC.

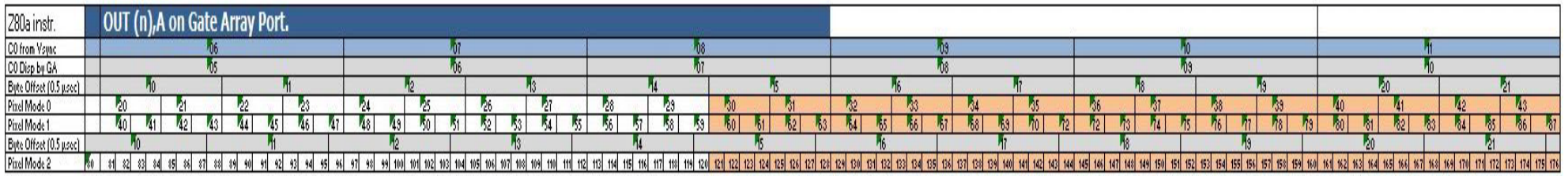
ATTENTION :
Il ne faut pas perdre de vue que c'est l'affichage des pixels en mode 2 qui débute plus tôt que pour les autres modes. La mise à jour de la couleur d'une encre, quant à elle, est prise en compte de manière identique quelque soit le mode graphique.

La couleur change donc sur le 2ème pixel d'un octet en mode 2, mais bien sur les pixels 0 des octets traités pour les autres modes graphiques.

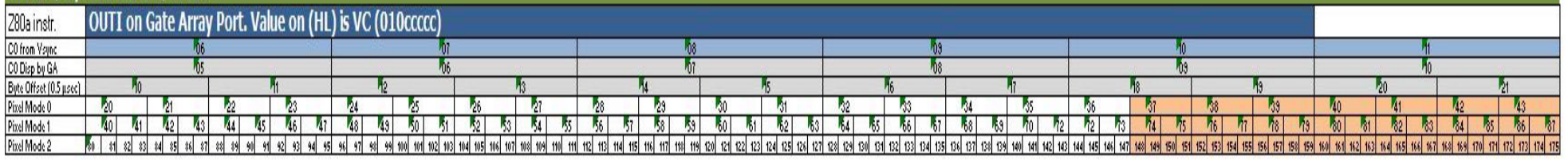
CRTC 0, 1, 2 / GATE ARRAY 40010

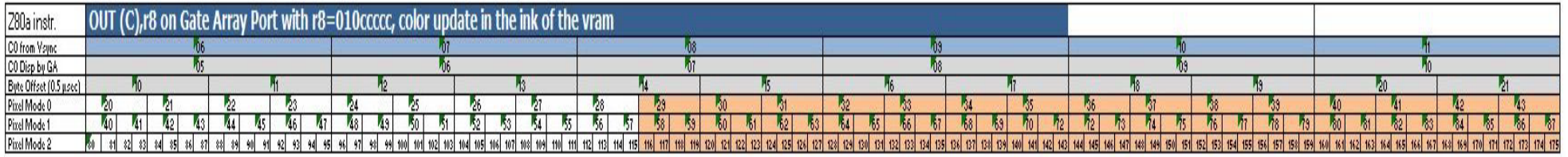


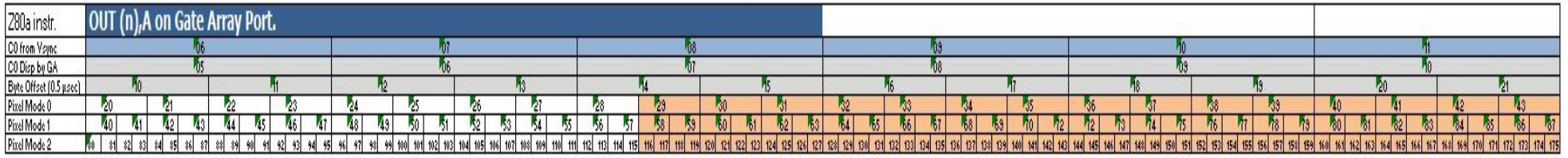




CRTC 3, 4 / ASIC 40489, 40226







9.4 MODE GRAPHIQUE

On peut considérer qu'une HSYNC indiquée par le CRTC est composée de plusieurs périodes au sein du GATE ARRAY.

Une période, qu'on peut nommer HSYNC-GA, a une durée maximale plafonnée à 6 µsec et ne peut pas dépasser la valeur définie dans R3.

Pour tous les CPC sans exception (CRTC 0 à 4) il faut une HSYNC de 2 µsec minimum pour que le changement de mode soit pris en compte. Il convient ici de noter qu'il est possible de réduire, sur les CRTC 0, 1 et 2, la zone de « non affichage » (traduite par un « noir profond ») de cette période à 1.6 µsec afin de réduire l'impact visuel d'une telle HSYNC générée dans la zone visible (voir chapitre 15, page 92). (mais qui s'amuserait à faire une chose pareille ?).

La mise à jour du registre interne contenant le mode est effective sur la 3ème µseconde de l'instruction Z80A d'I/O OUT [C],r8 à destination du GATE ARRAY.

La période de "latence" minimum pour que le GATE ARRAY génère le signal HSYNC nécessaire au moniteur est de 3 µsec.

Si la HSYNC-GA est limitée à 2 µsec (via la programmation de R3 du CRTC), alors le signal HSYNC n'est pas généré pour le moniteur. Dans cette situation, une HSYNC de 2 µsec n'incite pas le moniteur à rendre ses tripes.

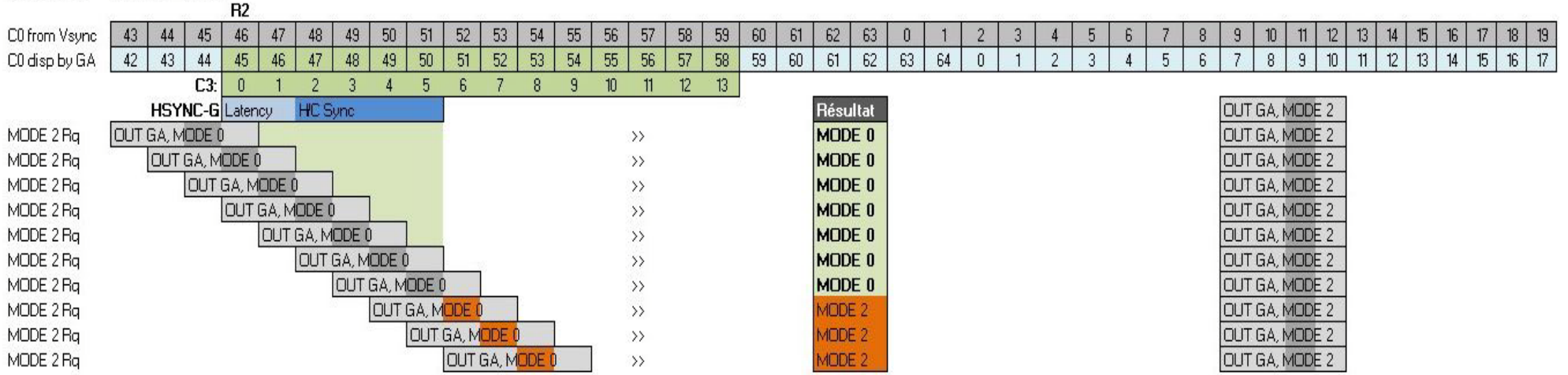
La HSYNC est prise en compte plus rapidement par le GATE ARRAY sur les CRTC 0, 1 et 2, que sur les ASIC des CRTC 3 et 4. Les ASIC **diffèrent la HSYNC de 1 µsec, comme ils le font pour l'affichage des caractères**.

Sur le principe du point de référence VSYNC, les schémas suivants décrivent **la prise en compte d'un changement de mode** par une instruction Z80A OUT(C),r8 avant et pendant la HSYNC.

CRTC 0, 1, 2

CRTC-R2=46

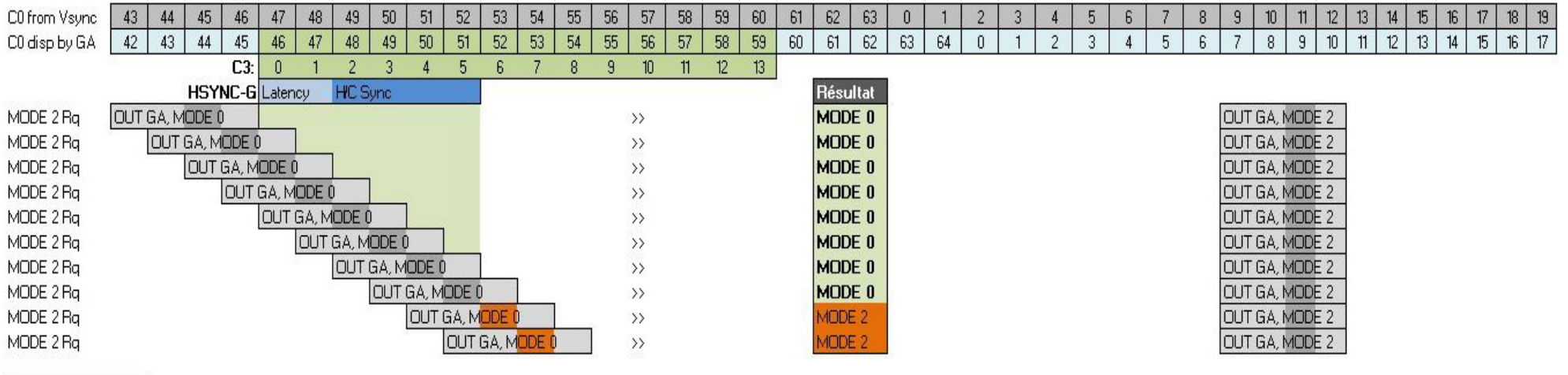
CRTC-R3=14 HBL Size=14 chars



CRTC 3, 4

CRTC-R2=46

CRTC-R3=14 HBL Size=14 chars



10 COMPTAGES : REGISTRE R9

10.1 GÉNÉRALITÉS

Le registre 9 permet de fixer le nombre de lignes verticales de chaque ligne-caractère.

En principe le compteur C9, qui détermine la ligne-raster de la ligne caractère, s'incrémente jusqu'à la valeur de R9.

Lorsqu'il revient à 0, C4 est incrémenté (et en "principe" doit revenir à 0 lorsque C4=R4).

Le compteur interne de "lignes" C9 est relié directement aux bits 11.12.13 du pointeur de VRAM.

C9 délimite, dans un espace de 16 ko, 8 tranches de 2 ko pour 8 "lignes" différentes possibles.

Etendu à l'espace de 64k adressable, cela représente 8 tranches de 8 ko pour les 8 « lignes ».
Ce principe participe à la non linéarité verticale de l'affichage (effet "rideau" lorsque la mémoire est transférée sans découpage) bien connu de tout habitué du CPC.

La définition de R9 est de 5 bits. On dispose de « caractères verticaux » de 32 lignes maximum. Les bits 3 et 4 du compteur ne sont pas pris en compte pour le calcul du pointeur vidéo sur les valeurs de C9 qui dépassent 7.

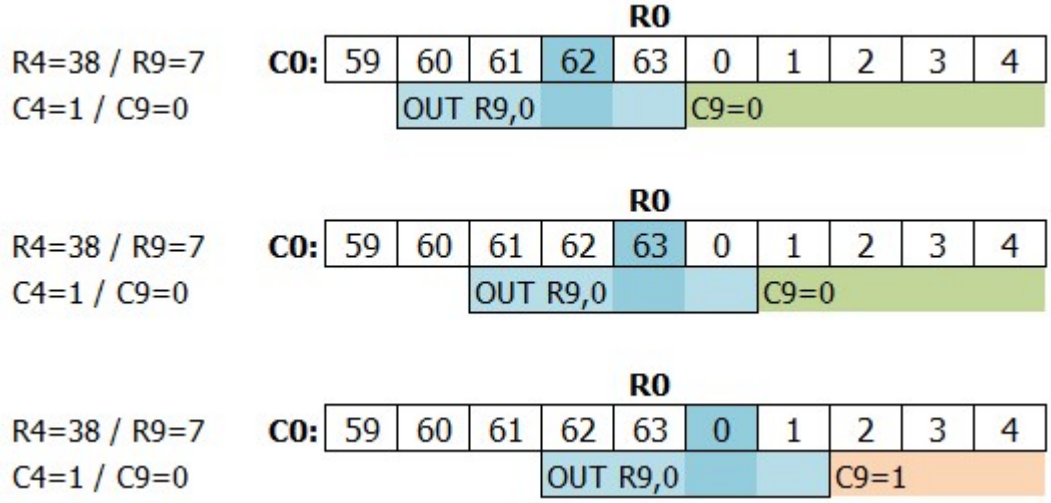
C'est néanmoins une information dont il faut tenir compte si on amène C9 à compter au-delà de 7. En effet, même si C9=8 donne l'affichage d'une ligne 0 (le bit 3 étant "ignoré"), cette valeur ne permet pas de prendre en compte les opérations qui ont lieu lors du changement de ligne-caractère (C4).

10.2 DELAIS DE PRISE EN COMPTE

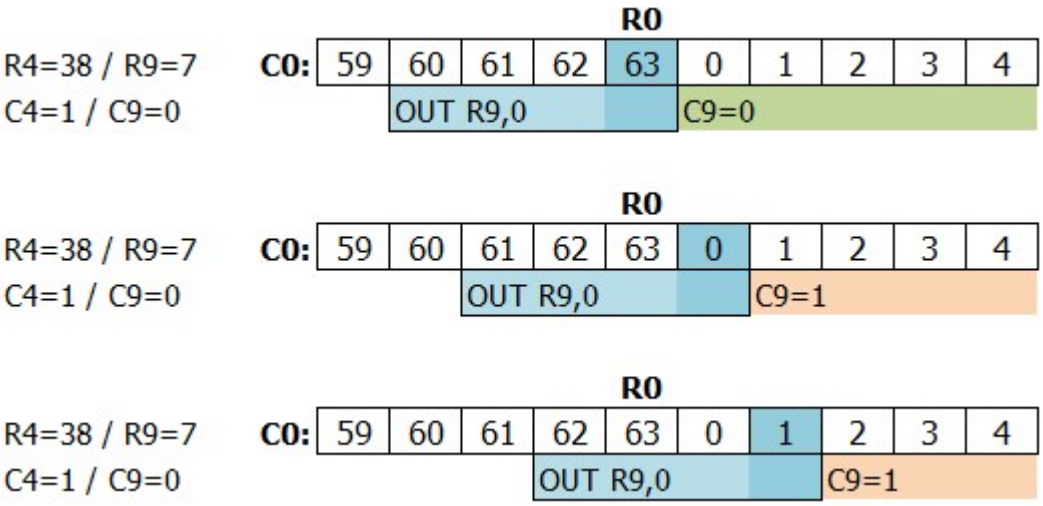
La mise à jour de R9 est prise en compte tant que C0<=R0.

Dans les schémas ci-dessous, R9 est mis à 0 alors qu'il était supérieur à 0 avant.

CRTC 0 , 1, 2



CRTC 3, 4



10.3 REGLES DE COMPTAGE

Si R9 est mis à jour avec 0 alors que C9 est supérieur à 0, par exemple, C9 va compter jusqu'à sa valeur maximale (31) avant de repasser à 0.

Il existe une exception à cette règle pour les CRTC 0 et 2, lorsque R9 est mis à jour avec 0 sur la dernière ligne du dernier caractère de l'écran.

10.3.1 CRTC 0

On peut lire assez souvent que la modification de R9 sur les CRTC 0 (et 2) n'est pas prise en compte pour la ligne qui suit celle ou la mise à jour a eu lieu car le compteur pour ce registre serait "bufférisé". On peut toutefois considérer que le pointeur vidéo est « bufferisé » car il est mis de côté à chaque début de ligne (voir chapitres 17 et 20.2).

Le CRTC ne "bufférise" pas ses compteurs. **Il teste simplement l'équivalence des compteurs avec ses registres à des instants très précis.**
En l'occurrence, la valeur de C9 est comparée avec R9 lorsque C0=0 (ainsi que C4 avec R4) pour déterminer si la ligne était la dernière de l'écran.

Cette comparaison a lieu uniquement pour décider si C4 sera remis à 0 ou si C4 sera incrémenté.

En effet, C4 est un compteur qui doit "officiellement" pouvoir dépasser R4, notamment en ajustement vertical.

Note : Les CRTC 3 et 4, pour lesquels C4 ne dépasse pas R4 en ajustement, "solidarise" les lignes additionnelles avec C4=R4.

Si R9 ou R4 sont modifiés après que C0 ait dépassé 0, cela ne change rien. L'affaire est pliée.
Lorsque C4 est incrémenté, C9 passe obligatoirement à 0 (sauf lorsque R0=0).

En dehors de cette gestion particulière de traitement de C4 sur la dernière ligne d'un écran, **la mise à jour de R9 est prise en compte immédiatement**.

10.3.1.1 CAS GENERAL

Si R9 est modifié avec la valeur de C9, il va passer à 0 sur la ligne suivante.
Exemple : C9=0 et on positionne R9=0, qui valait 7 avant.
Exemple : C9=3 et on positionne R9=3, qui valait 7 avant.

Si R9 est modifié avec une valeur inférieure à C9, alors C9=C9+1 (débordement de C9)
Exemple : C9=3 et on positionne R9=1. C9 vaudra 4.
Si le registre C9 déborde, il va compter jusqu'à sa valeur maximale (31) avant de reboucler à 0.

Si R9 est modifié avec une valeur supérieure à C9, alors C9=C9+1. C4 est inchangé.
Exemple : C9=0 et on positionne R9=7. C9 vaudra 1 sur la prochaine ligne.

10.3.1.2 EXCEPTION AU CAS GENERAL : DERNIERE LIGNE ECRAN

La mise à jour de R9 sur cette dernière ligne lorsque C0>1 n'a aucune incidence sur le prochain C9, car C4 sera forcé à 0 et donc C9 passera aussi à 0.

Si C4=R4 et C9=R9 lorsque C0=0, alors un état "dernière ligne écran" est levé.
Dans cette situation, C4 sera remis à 0 (hors ligne additionnelle), et C9 passera à 0 quelque soit la valeur programmée dans R9 ensuite.

Si C4=R4 et C9<>R9 quand C0=0, alors C4 sera incrémenté quoi qu'il arrive.
Dans cette situation, si R9 est modifié sur cette ligne avec la valeur de C9, alors C9 passera à 0 mais C4 sera incrémenté.

Si R9 est modifié sur une dernière ligne et que l'objectif est que l'état « dernière ligne » soit levé (afin que C4 repasse à 0), il faut que le CRTC puisse déterminer qu'il est sur une dernière ligne.
Il ne pourra le faire que lorsque C0 aura atteint 2 (voir chapitre 13.2).

- Si une mise à jour de R9 a lieu lorsque C0=0, alors l'état dernière ligne ne sera pas levé.
- Si une mise à jour de R9 a lieu lorsque C0=1, alors la gestion additionnelle a lieu et C9=R9 n'est plus géré jusqu'à ce que C9+1=R5.

10.3.2 CRTC 1

10.3.2.1 CAS GENERAL

Si R9 est modifié avec la valeur courante de C9, alors sur la ligne suivante :
C9 passe à 0.
C4 est incrémenté, et passe à 0 si il valait R4, et dans ce cas l'offset est pris en compte.

Si R9 est modifié avec une valeur inférieure à C9
C9=C9+1 et C4 est inchangé (jusqu'à la fin du débordement de C9).
L'offset sera modifié uniquement si C4=C9=C0=0

Si R9 est modifié avec une valeur supérieure à C9, alors C9=C9+1. C4 est inchangé.
Exemple : C9=0 et on positionne R9=7. C9 vaudra 1 sur la prochaine ligne.

10.3.2.2 AUCUNE EXCEPTION

Tout n'est que pure logique, dans une indicible et insipide simplicité. :-)

10.3.3 CRTC 2

10.3.3.1 CAS GENERAL

Si R9 est modifié avec la valeur de C9, il va passer à 0 sur la ligne suivante.
Exemple : C9=0 et on positionne R9=0, qui valait 7 avant
Exemple : C9=3 et on positionne R9=3, qui valait 7 avant

Si R9 est modifié avec une valeur inférieure à C9, alors C9=C9+1 (débordement de C9)
Exemple : C9=3 et on positionne R9=1. C9 vaudra 4.
Si le registre C9 déborde, il va compter jusqu'à sa valeur maximale (31) avant de reboucler à 0.

Si R9 est modifié avec une valeur supérieure à C9, alors C9=C9+1. (C4 est inchangé).
Exemple : C9=0 et on positionne R9=7. C9 vaudra 1 sur la prochaine ligne.

10.3.3.2 EXCEPTIONS AU CAS GENERAL

Lorsque C4=R4 et C9=R9 **durant la première HSYNC**, un état "dernière ligne écran" est levé, qui ne peut plus être annulé. Le test de dernière ligne a lieu sur chaque position de C0>0. Si R9 est modifié sur C0=0 lorsque C4=R4 et C9=R9, alors l'état dernière ligne n'est pas activé.

La mise à jour de R9 dans cette situation n'a aucune incidence sur le prochain C9, car **C4 sera forcé à 0** et C9 passera aussi à 0.

Lorsque la remise à 0 de C4 est armée, il n'est plus possible de l'annuler, sauf si la condition de dernière ligne est encore vraie **avec le même C9/R9 durant la seconde HSYNC**. **Dans ce cas, la remise à 0 de C4 est annulée de manière irrévocable**.
Si C9 valait R9, alors C4 sera incrémenté quelque soit la valeur de R4 (même si C4=R4).

Si R9 a été modifié (C9<>R9) alors C9 va s'incrémenter (C4 ne sera pas modifié).

Si l'objectif est de maintenir la remise à 0 de C4, il est possible d'empêcher l'armement de l'incrémentation irrévocable de C4/C9 décrite ci-dessus en changeant la condition de dernière ligne pour les valeurs de C0 durant lesquelles la HSYNC a lieu.
Voir RLAL/RVLL au chapitre 12.4.1, page 57.

10.3.4 CRTC 3, 4

10.3.4.1 CAS GENERAL

Lorsque R9 est modifié, sa valeur est prise en compte immédiatement de la manière suivante.
Si R9 est modifié avec une valeur inférieure ou égale à C9, alors C9 passe à 0 sur la ligne suivante, et C4 passe à 0 (si C4=R4) sinon C4=C4+1.

Dis autrement, il est impossible de faire "déborder" C9 sur ces CRTC.

Ce n'est pas un simple test d'égalité qui a lieu mais une comparaison « plus complexe » permise par un ASIC :
Si C9 courant > R9 alors C9 suivant=0
Exemple : Si C9 valait 4, et que R9 est modifié avec 1 (alors qu'il valait 7 auparavant), alors C9 va passer à 0 (et C4=C4+1 ou 0 selon la valeur de C4 et R4)

Après C9=0, C9 sera incrémenté jusqu'à la nouvelle valeur de R9. 1 dans l'exemple.

Cette gestion permet parfois une double compatibilité avec les CRTC 0, 1 et 2.

En effet, si un programme modifie R9 dans le cadre d'une rupture ligne à ligne, une manière classique de procéder est de :

- Positionner R9=0 sur la dernière ligne d'un écran sur CRTC 0 ou 2, afin que la prochaine ligne C9=0 soit aussi considérée comme le dernier écran (si R4=0).
- Positionner R9=0 sur la première ligne d'un écran sur CRTC 1, afin que la prochaine ligne soit considérée comme le dernier écran (si R4=0).

Dans ces deux situations, le CRTC 3 et 4 remettront C9=0 sur la prochaine ligne.

Previous R9=7 C4=R4			Event	CRTC 0 (HD6845SP) Result on next line			CRTC 1 (UM6845R) Result on next line			CRTC 2 (MC6845P) Result on next line			CRTC 3, 4 Result on next line		
Case	**R9**	**C9 cur**	**Upd R9**	**C9**	**C4**	**Offs Upd**	**C9**	**C4**	**Offs Upd**	**C9**	**C4**	**Offs Upd**	**C9**	**C4**	**Offs Upd**
1	7	0	0	0	**C4=C4+1 (*)**	No	0	0	Yes	0	**0**	**If C0<=R1**	0	0	Yes
2	7	1	0	C9=C9+1 (2)	unmodified	No	C9=C9+1 (2)	unmodified	Yes if C4=0	C9=C9+1 (2)	unmodified	No	0	0	Yes
3	7	2	0	C9=C9+1 (3)	unmodified	No	C9=C9+1 (3)	unmodified	Yes if C4=0	C9=C9+1 (3)	unmodified	No	0	0	Yes
4	7	3	0	C9=C9+1 (4)	unmodified	No	C9=C9+1 (4)	unmodified	Yes if C4=0	C9=C9+1 (4)	unmodified	No	0	0	Yes
5	7	4	0	C9=C9+1 (5)	unmodified	No	C9=C9+1 (5)	unmodified	Yes if C4=0	C9=C9+1 (5)	unmodified	No	0	0	Yes
6	7	5	0	C9=C9+1 (6)	unmodified	No	C9=C9+1 (6)	unmodified	Yes if C4=0	C9=C9+1 (6)	unmodified	No	0	0	Yes
7	7	6	0	C9=C9+1 (7)	unmodified	No	C9=C9+1 (7)	unmodified	Yes if C4=0	C9=C9+1 (7)	unmodified	No	0	0	Yes
8	7	7	0	0	0	Yes	C9=C9+1 (8)	unmodified	Yes if C4=0	0	**0**	**If C0<=R1**	0	0	Yes

(*) overflow

Previous R9=7 C4<>R4			Event	CRTC 0 (HD) Result on next line			CRTC 1 Result on next line			CRTC 2 Result on next line			CRTC 3, 4 Result on next line		
Case	**R9**	**C9 cur**	**Upd R9**	**C9**	**C4**	**Offs Upd**	**C9**	**C4**	**Offs Upd**	**C9**	**C4**	**Offs Upd**	**C9**	**C4**	**Offs Upd**
1	7	0	0	0	**C4=C4+1**	No	0	C4=C4+1	No	0	C4=C4+1	No	0	C4=C4+1	No
2	7	1	0	C9=C9+1 (2)	unmodified	No	C9=C9+1 (2)	unmodified	Yes if C4=0	C9=C9+1 (2)	unmodified	No	0	C4=C4+1	No
3	7	2	0	C9=C9+1 (3)	unmodified	No	C9=C9+1 (3)	unmodified	Yes if C4=0	C9=C9+1 (3)	unmodified	No	0	C4=C4+1	No
4	7	3	0	C9=C9+1 (4)	unmodified	No	C9=C9+1 (4)	unmodified	Yes if C4=0	C9=C9+1 (4)	unmodified	No	0	C4=C4+1	No
5	7	4	0	C9=C9+1 (5)	unmodified	No	C9=C9+1 (5)	unmodified	Yes if C4=0	C9=C9+1 (5)	unmodified	No	0	C4=C4+1	No
6	7	5	0	C9=C9+1 (6)	unmodified	No	C9=C9+1 (6)	unmodified	Yes if C4=0	C9=C9+1 (6)	unmodified	No	0	C4=C4+1	No
7	7	6	0	C9=C9+1 (7)	unmodified	No	C9=C9+1 (7)	unmodified	Yes if C4=0	C9=C9+1 (7)	unmodified	No	0	C4=C4+1	No
8	7	7	0	C9=C9+1 (8)	unmodified	No	C9=C9+1 (8)	unmodified	Yes if C4=0	C9=C9+1 (8)	unmodified	No	0	C4=C4+1	No

Previous R9=7 C4=R4			Event	CRTC 0 (HD) Result on next line			CRTC 1 Result on next line			CRTC 2 Result on next line			CRTC 3, 4 Result on next line		
Case	**R9**	**C9 cur**	**Upd R9**	**C9**	**C4**	**Offs Upd**	**C9**	**C4**	**Offs Upd**	**C9**	**C4**	**Offs Upd**	**C9**	**C4**	**Offs Upd**
1	7	0	0	0	**C4=C4+1**	No	0	0	Yes	0	**0**	**If C0<=R1**	0	0	Yes
2	7	1	1	0	C4=C4+1	No	0	0	Yes	0	**0**	**If C0<=R1**	0	0	Yes
3	7	2	2	0	C4=C4+1	No	0	0	Yes	0	**0**	**If C0<=R1**	0	0	Yes
4	7	3	3	0	C4=C4+1	No	0	0	Yes	0	**0**	**If C0<=R1**	0	0	Yes
5	7	4	4	0	C4=C4+1	No	0	0	Yes	0	**0**	**If C0<=R1**	0	0	Yes
6	7	5	5	0	C4=C4+1	No	0	0	Yes	0	**0**	**If C0<=R1**	0	0	Yes
7	7	6	6	0	C4=C4+1	No	0	0	Yes	0	**0**	**If C0<=R1**	0	0	Yes
8	7	7	7	0	0	Yes	0	0	Yes	0	**0**	**If C0<=R1**	0	0	Yes

Previous R9=7 C4<>R4			Event	CRTC 0 (HD) Result on next line			CRTC 1 Result on next line			CRTC 2 Result on next line			CRTC 3, 4 Result on next line		
Case	R9	C9 cur	Upd R9	C9	C4	Offs Upd	C9	C4	Offs Upd	C9	C4	Offs Upd	C9	C4	Offs Upd
1	7	0	0	0	**C4=C4+1**	No	0	C4=C4+1	No	0	C4=C4+1	No	0	C4=C4+1	No
2	7	1	1	0	C4=C4+1	No	0	C4=C4+1	No	0	C4=C4+1	No	0	C4=C4+1	No
3	7	2	2	0	C4=C4+1	No	0	C4=C4+1	No	0	C4=C4+1	No	0	C4=C4+1	No
4	7	3	3	0	C4=C4+1	No	0	C4=C4+1	No	0	C4=C4+1	No	0	C4=C4+1	No
5	7	4	4	0	C4=C4+1	No	0	C4=C4+1	No	0	C4=C4+1	No	0	C4=C4+1	No
6	7	5	5	0	C4=C4+1	No	0	C4=C4+1	No	0	C4=C4+1	No	0	C4=C4+1	No
7	7	6	6	0	C4=C4+1	No	0	C4=C4+1	No	0	C4=C4+1	No	0	C4=C4+1	No
8	7	7	7	0	C4=C4+1	No	0	C4=C4+1	No	0	C4=C4+1	No	0	C4=C4+1	No

Previous R9=7 C4=R4			Event	CRTC 0 (HD) Result on next line			CRTC 1 Result on next line			CRTC 2 Result on next line			CRTC 3, 4 Result on next line		
Case	R9	C9 cur	Upd R9	C9	C4	Offs Upd	C9	C4	Offs Upd	C9	C4	Offs Upd	C9	C4	Offs Upd
1	7	0	2	C9=C9+1 (1)	unmodified	No	C9=C9+1 (1)	unmodified	Yes if C4=0	C9=C9+1 (1)	unmodified	No	0	0	Yes

Previous R9=0 C4=R4=0			Event	CRTC 0 (HD) Result on next line			CRTC 1 Result on next line			CRTC 2 Result on next line			CRTC 3, 4 Result on next line		
Case	R9	C9 cur	Upd R9	C9	C4	Offs Upd	C9	C4	Offs Upd	C9	C4	Offs Upd	C9	C4	Offs Upd
1	0	0	0	0	0	Yes	0	0	Yes	**0 (*)**	**C4=0 / C4+1**	Yes (C0<=R1)/No	0	0	Yes
2	0	0	1	0	0	Yes	C9=C9+1 (1)	0	Yes (C4=0)	**0 / 1 (*)**	**C4=0 / C4+1**	Yes (C0<=R1)/No	C9=C9+1 (1)	0	No
3	0	0	2	0	0	Yes	C9=C9+1 (1)	0	Yes (C4=0)	**0 / 1 (*)**	**C4=0 / C4+1**	Yes (C0<=R1)/No	C9=C9+1 (1)	0	No
4	0	0	3	0	0	Yes	C9=C9+1 (1)	0	Yes (C4=0)	**0 / 1 (*)**	**C4=0 / C4+1**	Yes (C0<=R1)/No	C9=C9+1 (1)	0	No
5	0	0	4	0	0	Yes	C9=C9+1 (1)	0	Yes (C4=0)	**0 / 1 (*)**	**C4=0 / C4+1**	Yes (C0<=R1)/No	C9=C9+1 (1)	0	No
6	0	0	5	0	0	Yes	C9=C9+1 (1)	0	Yes (C4=0)	**0 / 1 (*)**	**C4=0 / C4+1**	Yes (C0<=R1)/No	C9=C9+1 (1)	0	No
7	0	0	6	0	0	Yes	C9=C9+1 (1)	0	Yes (C4=0)	**0 / 1 (*)**	**C4=0 / C4+1**	Yes (C0<=R1)/No	C9=C9+1 (1)	0	No
8	0	0	7	0	0	Yes	C9=C9+1 (1)	0	Yes (C4=0)	**0 / 1 (*)**	**C4=0 / C4+1**	Yes (C0<=R1)/No	C9=C9+1 (1)	0	No

(*) According cancelation of C4 zeroed in Hsync - See chapter

11 COMPTAGES : REGISTRE R5

11.1 GÉNÉRALITÉS

Le registre R5 permet d'ajouter des lignes dites d'ajustement vertical en fin d'écran.
L'objectif de ce registre est de permettre de compléter le nombre total de lignes affichées lorsque (R4+1)x(R9+1) appliqué durant un frame ne correspond pas à 312 lignes (pour le standard Européen).

Ce registre contient un nombre de lignes sur 5 bits (0 à 31) qui correspond à l'ajustement maximum possible par rapport au nombre maximum de lignes possibles pour R9.

Lorsque C4=R4, C9=R9 et C0=R0, alors C5=0 et C9=0.

Si R5 = 0 alors il n'y pas d'ajustement spécifique.
Si R5 <> 0 alors un ajustement a lieu avec création de lignes complémentaires.

L'implémentation de la fonction d'ajustement vertical est source de plusieurs différences entre les CRTC, avec des conséquences plus ou moins heureuses. Durant cette création de lignes complémentaires, les compteurs C5 et C9 sont incrémentés, et parfois C4 :

- Sur le CRTC 0, C4 s'incrémente 1 seule fois étant donné que C9 fait office de C5.
- Sur les CRTC 1 et 2, C4 s'incrémente quelque soit la valeur de R4 à chaque fois que C9=R9.
- Sur les CRTC 3 et 4, C4 ne s'incrémente pas.

La gestion de R1 pour la mise à jour du pointeur vidéo continue à être assurée durant cette gestion de lignes additionnelles sur les CRTC 0, 1, 2 lorsque C9=R9.

La fin de gestion additionnelle, lorsque C5=R5 (ou C9+1=R5 sur CRTC 0) provoque la remise à 0 de C4.

R7 peut être positionné sur une des valeurs atteinte par C4 en ajustement vertical pour déclencher une VSYNC.

R6 doit être positionné en fonction des valeurs atteintes par C4 pour afficher les caractères correspondants.

11.2 COMPTAGE EN AJUSTEMENT VERTICAL

11.2.1 GÉNÉRALITÉS

Les schémas suivants décrivent la diversité des méthodes de comptage (C4, C9, C5) et de mise à jour du pointeur vidéo durant cette gestion, en considérant que R5 et R9 ne sont pas modifiés durant l'ajustement. Les registres sont initialisés ainsi (R4=10, R5=16, R9=3, R1=40, R0=63)

CRTC 0				
C4	C9	*C5*	PTR-VRAM	LINE
11	0	*0*	0	&0
11	1	*1*	0	&800
11	2	*2*	0	&1000
11	3	*3*	0	&1800
11	4	*4*	40	&2000
11	5	*5*	40	&2800
11	6	*6*	40	&3000
11	7	*7*	40	&3800
11	8	*8*	40	&0
11	9	*9*	40	&800
11	10	*10*	40	&1000
11	11	*11*	40	&1800
11	12	*12*	40	&2000
11	13	*13*	40	&2800
11	14	*14*	40	&3000
11	15	*15*	40	&3800

CRTC 1, 2				
C4	C9	C5	PTR-VRAM	LINE
11	0	0	0	&0
11	1	1	0	&800
11	2	2	0	&1000
11	3	3	0	&1800
12	0	4	40	&0
12	1	5	40	&800
12	2	6	40	&1000
12	3	7	40	&1800
13	0	8	80	&0
13	1	9	80	&800
13	2	10	80	&1000
13	3	11	80	&1800
14	0	12	120	&0
14	1	13	120	&800
14	2	14	120	&1000
14	3	15	120	&1800

CRTC 3, 4				
C4	C9	C5	PTR-VRAM	LINE
10	0	0	0	&0
10	1	1	0	&800
10	2	2	0	&1000
10	3	3	0	&1800
10	0	4	0	&2000
10	1	5	0	&2800
10	2	6	0	&3000
10	3	7	0	&3800
10	0	8	0	&0
10	1	9	0	&800
10	2	10	0	&1000
10	3	11	0	&1800
10	0	12	0	&2000
10	1	13	0	&2800
10	2	14	0	&3000
10	3	15	0	&3800

11.2.2 CRTC 0

Sur CRTC 0, il semble que les ingénieurs HITACHI aient fait l'économie d'un compteur C5 afin d'utiliser C9 en lieu et place. La nouvelle limite de C9 n'est alors plus R9 mais R5.

La gestion additionnelle inhibe alors simplement la remise à 0 de C9.
Ce n'est pas « pratique » pour gérer plusieurs caractères automatiquement durant l'ajustement, mais cela permet de passer sur une autre adresse sans que C9 soit égal à 0.

En effet, la gestion C9=R9 continue de prendre en compte le pointeur vidéo (VMA'=VMA) lorsque C0=R1. Ainsi, en reprenant les données du premier schéma, la mise à jour de R9 en cours d'ajustement sera prise en compte pour faire évoluer le pointeur VMA'.

CRTC 0					
C4	C9	*C5*	PTR-VRAM	LINE	
11	0	*0*	0	&0	
11	1	*1*	0	&800	
11	2	*2*	0	&1000	
11	**3**	*3*	0	&1800	**R9**
11	4	*4*	40	&0	**OUT R9,10**
11	5	*5*	40	&800	
11	6	*6*	40	&1000	
11	7	*7*	40	&1800	
11	8	*8*	40	&0	
11	9	*9*	40	&800	
11	**10**	*10*	40	&1000	**R9**
11	11	*11*	80	&1800	
11	12	*12*	80	&0	
11	13	*13*	80	&800	
11	14	*14*	80	&1000	
11	15	*15*	80	&1800	

11.2.3 CRTC 1, 2

Sur ces circuits, les compteurs C5 et C9 sont dissociés.
La gestion C9=R9 prend en compte le pointeur vidéo (VMA'=VMA) et incrémente C4 lorsque C0=R1, mais remet C9 à 0 lorsque C9=R9.

La mise à jour de R9 est prise en compte pour le comptage courant de C9.

CRTC 1, 2					
C4	C9	C5	PTR-VRAM	LINE	
11	0	0	0	&0	
11	1	1	0	&800	
11	2	2	0	&1000	
11	3	3	0	&1800	**R9**
12	0	4	40	&0	**OUT R9,10**
12	1	5	40	&800	
12	2	6	40	&1000	
12	3	7	40	&1800	
12	4	8	40	&2000	
12	5	9	40	&2800	
12	6	10	40	&3000	
12	7	11	40	&3800	
12	8	12	40	&0	
12	9	13	40	&800	
12	10	14	40	&1000	**R9**
13	0	15	80	&0	

11.2.4 CRTC 3, 4

Sur les CRTC 3 et 4, le compteur C4 n'est pas incrémenté.
Cela "solidarise" le dernier caractère avec le caractère généré en ajustement vertical.

Dans l'exemple précédent, le caractère C4=10 contient 16 lignes de plus.

Les bits 10.11.12 du pointeur proviennent des bits 0.1.2 de C5 (et non de C9).

11.3 MISE A JOUR DE R5 DURANT UN AJUSTEMENT

Si R5 est modifié avec C5+1 sur la ligne courante, alors l'ajustement vertical est "stoppé".

Dans cette situation, C4=C9=0 pour la prochaine ligne (quelque soit la valeur courante de C9).
L'état « dernière ligne » est alors vrai et conduit à la remise à 0 de C4.

Cependant, si R5 est modifié (sournoisement ?) avec la valeur 0, alors C4 n'est pas remis à 0 et cela provoque un débordement C4.

L'algorithme d'ajustement vertical est alors désactivé, et C4 compte jusqu'à sa valeur maximum pour reboucler.

11.4 PRISE EN COMPTE AJUSTEMENT

CRTC 0

Sur le CRTC 0, la désactivation de la gestion C9=R9 en ajustement laisse supposer que C5 n'existe pas sur ce circuit. Ainsi, comme indiqué dans le chapitre dédié au registre R0, si C0 n'atteint pas 2 sur une dernière ligne d'écran, alors la gestion additionnelle est activée.
Une seule ligne est affichée car si C9 repasse à 0, il atteint R5=0.

En revanche, si C9 a eu le temps de s'incrémenter (cas du débordement documenté au chapitre 13.6.2.2) alors la gestion additionnelle se poursuit tant que C5 <> R5.

11.5 AJUSTEMENTS EN FOLIE...

CRTC 1

Le comportement suivant a été constaté sur le CRTC 1 uniquement

Lorsque R5 est modifié quand C0=R0, quelque soit la ligne, cela provoque :

- Un défaut de mise à jour du pointeur vidéo VMA'.
- Un traitement d'ajustement vertical.
- Un blocage de la mise à jour de VMA' lorsque C0=R1 et C9=R9

L'intégralité des compteurs et registres qui participent à la synchronisation et la construction de l'image ne sont pas affectés (dans le test).

Si le compteur C5 déborde, il va générer exactement 32 lignes à partir de la ligne suivant le bug. Cependant, le pointeur vidéo est corrompu.

Lorsque l'ajustement vertical cesse, VMA'=R12/R13, mais C4 ne repasse pas à 0 (si il n'a pas atteint R4) et l'écran reste donc synchronisé dans ce cas.

Cependant, le pointeur vidéo VMA n'est plus mis à jour avec VMA' lorsque C0=R1 et C9=R9, pour une raison inconnue.

Ce qui provoque une répétition des lignes caractères sur le reste de l'écran pour toutes les valeurs de C4 restant à afficher.

CRTC 0

Selon les documents HITACHI (CRTC 0) la mise à jour de R5 lorsque C0=R0 peut provoquer un défaut de gestion de R5 si d'autres conditions mystère sont remplies... Un hasard ?

11.6 R6 ET AJUSTEMENT VERTICAL

Lorsque C4 atteint R6, l'affichage des données est stoppé.
Sur le CRTC 1, il existe une gestion particulière de R6 lorsqu'il vaut 0, qui permet d'activer le BORDER de manière non définitive. Ce traitement particulier est géré en ajustement vertical.

11.7 AJUSTEMENT INTERLACIQUE

En mode INTERLACE, une gestion spécifique d'ajustement vertical est réalisée, avec l'ajout d'une ligne un écran sur deux. Voir chapitre 19.3, page 127.

Cet ajustement intervient pendant les écrans "pairs".

Le compteur de frame serait géré lorsque C4=R6. (tests à réaliser, mais non constaté)

La condition d'ajustement (mode interlace activé et frame pair) est évaluée sur la dernière ligne d'un écran, lorsque C0=R0, et uniquement si R8 contient la bonne valeur sur la dernière ligne.

Lorsque la condition d'ajustement est remplie, la ligne est ajoutée en dernier.

Cela peut être derrière la dernière ligne générée lorsque R5>0

Ou également derrière la ligne additionnelle générée par le CRTC 0 lorsque R0=1 et R5=0.
(à vérifier)

12 COMPTAGES : REGISTRE R4

12.1 GÉNÉRALITÉS

Le registre R4 du CRTC permet de définir le nombre de « ligne-caractères » à afficher.
Une ligne-caractère est composée de plusieurs lignes-raster, dont le nombre est fixé par le registre R9. Voir chapitre 10, page 42.

La ligne et le caractère sont numérotés à partir de 0.
Le nombre à programmer représente une valeur à atteindre.

Lorsque toutes les lignes d'un caractère sont affichées (C9=R9), alors C4 est incrémenté (ce dernier repasse à 0 si C4 valait R4 avant incrémentation).

La valeur de C4 est comparée avec R7 pour déclencher une VSYNC, et comparé avec R6 pour déclencher du BORDER.

Lorsque C4=R4 et que toutes les lignes du dernier caractère sont affichées, une gestion de ligne(s) additionnelle(s) peut éventuellement débuter si R5 > 0 ou que le mode Interlace est activé.

A noter qu'à l'exception des CRTC 3 et 4, les lignes additionnelles via R5 continuent à incrémenter C4 (qui dépasse donc R4 dans cette situation).

A noter également que cette gestion de ligne(s) additionnelle(s) est obligatoirement déclenchée lorsque R0<2 sur CRTC 0.

Lorsque C4 passe à 0 (et tant qu'il est à 0 sur CRTC 1), VMA est mis à jour avec le contenu de R12/R13.

Sur CRTC 2 cependant, VMA est mis à jour avec VMA', qui est lui-même mis à jour avec R12/R13 lorsque C0=R1 de la dernière ligne (lorsque C9=R9).

12.2 CRTC 0

Lorsque C0=0, le CRTC évalue si C9=R9 et C4=R4 pour **déterminer si il est sur la dernière ligne de l'écran**. Il ne refait plus ce test sur les autres valeurs de C0.

Il n'est pas forcément nécessaire d'anticiper la programmation de R4 sur la ligne précédente pour que C4=R4 au début de la ligne suivante. En positionnant un OUT R4 sur C0vs=#3E (si R0=#3F), R4 est mis à jour lorsque C0=0 pour satisfaire le test de "**dernière ligne**". Ceci permet une compatibilité avec les autres CRTC.

Lorsque le CRTC a déterminé qu'il est sur la dernière ligne de l'écran, C4 passera à 0 quoi qu'il arrive (ainsi que C9). Si R4 et/ou R9 sont modifiés en cours de ligne lorsque C0>1 alors que c'est **la dernière ligne, cela ne change rien pour C4 qui va donc passer à 0**.

Lorsque Le CRTC a déterminé qu'il n'était pas sur la dernière ligne (mais que R9 a été modifié avec C9), C4 va être incrémenté quoi qu'il arrive.

Si R4 et/ou R9 sont modifiés en cours de ligne alors que **ce n'est pas la dernière ligne, cela ne change rien pour C4 qui sera incrémenté** (lorsque C9=R9).

Lorsqu'on est sur la dernière ligne de l'écran :

- Modifier R4 avec C4 n'empêche pas C4 de s'incrémenter si C9=R9 (si l'objectif était de le faire passer à 0).
- Modifier R4 avec 0 alors que C4 valait R4 n'empêche pas C4 de revenir à 0 (si l'objectif était de le faire déborder).

Le test de "dernière ligne" est fait en début de ligne, aussi il est nécessaire d'anticiper la mise à jour de R4 et R9 en fonction de ce que l'on souhaite obtenir dans C4.

A noter que C9 peut uniquement déborder dans un contexte particulier (lorsque R0=0).

Le compteur C4 peut s'incrémenter au-delà de la valeur fixée dans R4 dans plusieurs cas de figure :

- Si R4 est mis à jour avec une valeur inférieure à C4 ou qu'il a été incrémenté dans les conditions décrites ci-dessus, alors C4 va "déborder". Autrement dit, il va s'incrémenter jusqu'à sa valeur maximale (127) avant de reboucler (si une nouvelle mise à jour de R4 n'intervient pas avant).
- C4 peut également dépasser la valeur de R4 lorsqu'une gestion de lignes verticales additionnelles est demandée. Cet évènement peut survenir lorsque R5>0, ou si le mode INTERLACE est activé sur des écrans pairs (voir chapitre 19.3, page 127), ou si R0<2.
  A noter que si C4 dépasse R4 sur au moins un de ces évènements, **il va revenir à 0 une fois la gestion additionnelle terminée**.

12.2.1 CAS PRATIQUE : RUPTURE LIGNE A LIGNE (R.L.A.L.)

L'objectif de cette technique est d'obtenir des lignes consécutives pour lesquelles C9=C4=0, afin de pouvoir en modifier l'adresse via R12 et/ou R13.
Pour les deux exemples suivants, on considère que les registres R4 et R9 sont supérieurs à 0.

A partir de la première ligne d'un écran :
Si R9 et/ou R4 sont positionnés à 0 lorsque C9=C4=0, cette ligne ne sera pas considérée comme la dernière de l'écran (le test a eu lieu lorsque C0=0).
Sur cette deuxième ligne, C4 va alors être incrémenté (C4=1). Et C9 va passer à 0.
Au début de cette deuxième ligne, C4(=1)<>R4(=0). C9=R9=0.
Le CRTC va donc incrémenter C4 quelque soit la valeur qu'on programme dans R4, ...

Si, sur la ligne 1, on programme R4=1 et R9=0 (au lieu de R4=0 et R9=0), les choses vont mieux se passer. Cette ligne n'ayant pas été considérée comme la dernière, C4 va s'incrémenter sur la deuxième ligne (C4=1)(et C9 sera égal à 0)

Cependant, le test qui a eu lieu en début de ligne 2 indique que c'était la dernière (C4=R4=1 et C9=R9=0). C4 va donc passer à 0 sur la troisième ligne.
Mais si on veut que cette troisième ligne et toutes les autres soient aussi considérées comme les dernières, il faut....
.... modifier R4 avec 0 sur la deuxième ligne. Et le tour est joué.

A partir de la dernière ligne d'un écran (C9=R9 et C4=R4):
Si R9 et R4 sont mis à jour avec 0 lorsque C9=R9 et C4=R4, cette ligne est déjà considérée comme la dernière. Sur la nouvelle ligne, C9 et C4 sont tous les deux passés à 0, mais pas à cause de la modification de R9 ou de R4 ou d'une quelconque « bufferisation ».

Au début de cette ligne entre C0=0 et 1, la condition C9=R9=0 et C4=R4=0 est vraie.
Cette ligne et les prochaines seront considérées comme les dernières.

Remarque : Lorsque C9 devient égal à R9 (dernière ligne), il faut attendre que C0=2 pour modifier R9, car le CRTC 0 a besoin des traitements particuliers en C0=0 et 1 pour gérer C9 et désactiver notamment la gestion additionnelle de lignes activée par défaut.

C4:	0																		
C9:	7																		
C0:	**0**	**1**	2	3	4	5	6	7	8	9	10	11	12	...	58	59	60	61	62	63
R9:	**7**	**7**	**0**	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
R4:	0	0	0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
	OUT R9,0														Update R12/R13 before C0=0					

Comment on sort ?
Pour arrêter d'obtenir des lignes C4=0 et C9=0, il est nécessaire de modifier R4 et/ou R9.
Etant donné que chaque ligne est aussi une "**dernière ligne**", C4 et R9 seront remis à 0 sur la ligne suivante, quelque soit la mise à jour de R4 et/ou R9.
(car C4 sera forcé à 0, et de facto, C9 aussi).

Dans la perspective d'un code commun avec le CRTC 2, il est conseillé de :

- Modifier R12/R13 avant que C0=R1.
- Gérer la mise à jour de R9 en HSYNC comme indiqué au chapitre 12.4.1.
- Ne pas positionner R2=0 (afin que le BORDER ne reste pas activé).

12.3 CRTC 1

Si R4 est mis à jour avec la valeur de C4 :

- Si on était sur la ligne C9 entre 0 et R9-1, alors C9=C9+1 (C4 passera à 0 lorsque C9 repassera à 0 et l'offset (R12/R13) sera pris en compte).
- Si on était sur la dernière ligne (C9=R9), alors C9 passe à 0, C4=0 et R12.R13 est pris en compte.

Si R4 est mis à jour avec une valeur inférieure à C4, alors C4 va s'incrémenter jusqu'à sa valeur maximale (127) avant de reboucler.

Contrairement au CRTC 0, mettre R4=0 sur la dernière ligne d'un écran va provoquer un "débordement" de C4, la valeur 0 étant gérée comme un cas général.

Si on veut mettre R4 à 0 pour que C4 boucle à 0, il faut donc le faire uniquement quand C4=0.

Créer une rupture ligne à ligne sur ce CRTC est trivial, car il suffit de mettre R4 et R9 à 0 lorsque C4 et C9 sont à 0.

12.4 CRTC 2

Le CRTC évalue, lorsque C0=0, si C9=R9 et C4=R4 pour déterminer **si il est sur la dernière ligne de l'écran**.

Lorsque cette condition est vraie, le CRTC arme une remise à 0 de C4 pour la prochaine ligne (C9 passe à 0 systématiquement lorsque C4 passe à 0). **Si R4 et/ou R9 sont de nouveau modifiés durant cette dernière ligne, cela ne change rien pour C4 et C9 qui vont repasser à 0**.

Un mécanisme sournois peut toutefois empêcher cette remise à 0 de C4.
Si une condition de dernière ligne se présente au début d'une ligne, elle est correctement gérée (remise à 0 inconditionnelle de C4 et de C9) pour cette ligne.

Cependant, si la condition « **dernière ligne** » est encore vraie **lorsque C0 repasse sur le dernier caractère de la HSYNC de la ligne suivante sans que C9 et R9 aient changé**, alors l'état dernière ligne est **désarmé** et C4 ne pourra pas repasser à 0.
Si R9 est modifié durant cette nouvelle « dernière ligne », C9 sera calculé en relation avec cette nouvelle valeur de R9.
Si R9 n'est pas modifié (C9=R9) alors C4 sera incrémenté inconditionnellement, quelque soit la valeur de R4.

Aussi, il suffit d'annuler la condition « **dernière ligne** » sur le dernier caractère de la HSYNC pour permettre à ce CRTC de gérer la remise à 0 de C4 comme attendu. **Modifier R9 judicieusement avant et après la HSYNC permet d'atteindre cet objectif**.

Lorsque le CRTC n'est pas sur la dernière ligne, la prise en compte de R4 est immédiate.

- C4 sera incrémenté si R9 est modifié en cours de ligne avec la valeur de C9.
- C4 reviendra à 0 si R4 est modifié en cours de ligne avec la valeur de C4 et que C9=R9.

12.4.1 CAS PRATIQUE : RUPTURE LIGNE A LIGNE (R.L.A.L.)

L'objectif de cette technique est que toutes les lignes affichées débutent avec C4=0 et C9=0, afin que les registres R12 et R13, modifiés avant que C0=R1, soient pris en compte.

Supposons que nous soyons sur la **dernière ligne** d'un écran avec C4=R4 et C9=R9 (R4 et/ou R9 sont supérieurs à 0). On sait que sur la prochaine ligne, C9=0 et C4=0, mais on souhaite que pour les prochaines lignes, C9 et C4 soient également à 0 (et pouvoir modifier l'adresse via R12 / R13).

Il sera nécessaire de reprogrammer R9 et R4 avec 0 afin que la condition de **dernière ligne** soit active sur chaque ligne. Chaque nouvelle ligne sera **une première et une dernière ligne** de l'écran.

Pour cet exemple, la HSYNC sera positionnée sur la position C0=1 (via R2=1).
Il faut tenir compte des contraintes liées aux autres défauts de gestion durant la HSYNC, comme la non-activation de la broche VSYNC et la désactivation du BORDER sur C0=0.

On va considérer que cette HSYNC aura le droit au temps minimum nécessaire à une synchro horizontale, à savoir 6 µsec (via R3=6) car il sera nécessaire de modifier 2 fois R9. Autant le faire au plus vite lorsque l'index sur le registre du CRTC est déjà sélectionné.

Si C9=C4=0 et que la condition de **dernière ligne** est vraie sur le dernier caractère de la HSYNC, alors l'incrémentation inconditionnelle de C4 est armée. En modifiant R9 durant la HSYNC afin que la condition de dernière ligne soit fausse, il est possible de désarmer cette incrémentation. En modifiant R9 avec une autre valeur que celle de C9, le traitement en HSYNC ne peut plus corrompre le traitement C4 car sa condition C9=R9 n'est plus satisfaite.

Une fois que la HSYNC est terminée, remettre la valeur de R9 identique à celle de C9 (donc 0), permet d'indiquer quelle est la bonne limite de C9, afin que la condition « dernière ligne » soit traitée correctement.

Exemple en image avec 3 lignes:

1ère ligne :

On suppose ici que C4=R4=0 et C9=R9=7, et la HSYNC est représentée en orange :

	R2													...					
C4:	0																		
C9:	**7**																		
C0:	0	1	2	3	4	5	6	7	8	9	10	11	12	...	58	59	60	61	62	63
C3:		1	2	3	4	5	6													**R1**
R9:	7	7	7	7	7	7	7	0	0	0	0	0	0	...	0	0	0	0	0	0
R4:	0	0	0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
						OUT R9,0					Update R12/R13 before C0=R1									

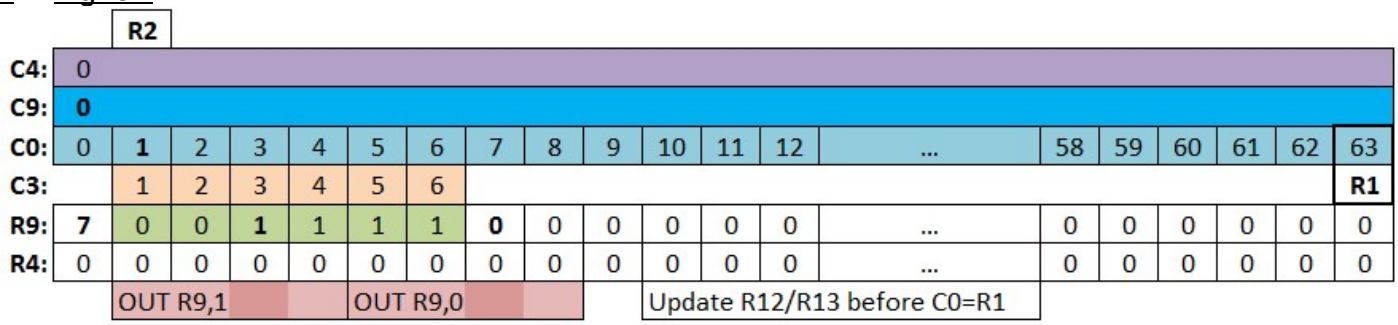
Cette ligne est considérée comme la dernière de l'écran car C9=R9 et C4=R4 lorsque C0=0. Cette condition vraie dans la HSYNC n'est pas prise en compte (pour annuler la remise à 0 de C4) car durant la HSYNC précédente, C9 valait 6 et R9 valait 7 (condition « dernière ligne » fausse). R12 et/ou R13 sont modifiés avant que C0=R1 afin que VMA' soit mis à jour avec R12/R13 car on est... sur la dernière ligne.

Remarque : Cette affectation de R12/R13 dépend de l'état « dernière ligne » lorsque C0=R1 (et non de l'égalité C4=R4 et C9=R9 qui positionne cet état). Ceci implique que la modification de R9 avant l'égalité C0=R1 n'empêche pas cette affectation.

Remarque : La modification de R9=0 sur cette ligne permet que la suivante soit considérée comme une dernière ligne.

Remarque : R9 doit être modifié à partir de C0=1 afin que la condition dernière ligne soit vraie sur C0=0. Si R9 est modifié sur C0=0, alors cette ligne n'est pas considérée comme la dernière (0 est différent de 7)(et donc C4 sera incrémenté).

2ème ligne :



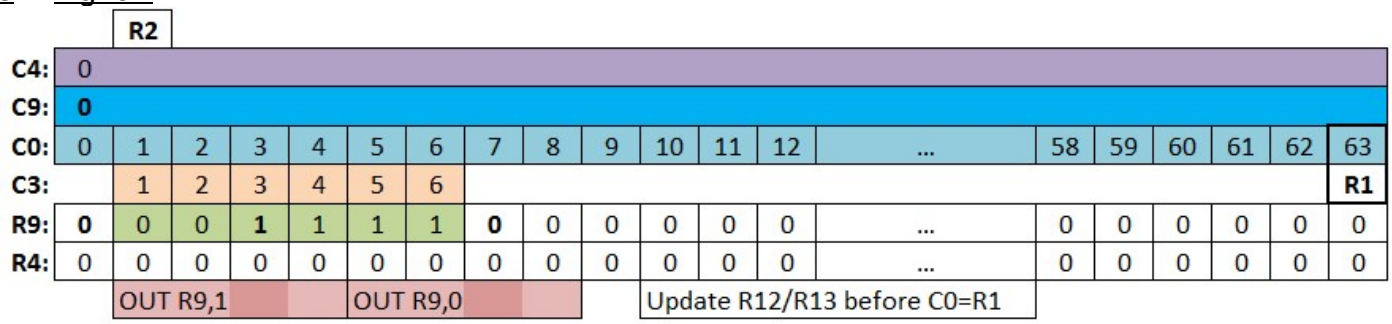
Sur C0=6 (dernier caractère de la HSYNC) le test C9(0)=R9(1) est faux.
L'incrémentation de C4 n'est pas armée car la condition requise n'est pas respectée.

Le OUT R9,0 a pour effet de modifier R9 lorsque C0=7 et que la HSYNC est terminée.
Sur C0=7, le CRTC arme la remise à 0 de C4 et de C9 (condition de « dernière ligne » satisfaite).

Remarque : Cette remise à 0 de C9 (et de C4) ne peut alors plus être « annulée ».
Si, dans l'exemple ci-dessus, un OUT R9,10 était ajouté derrière le OUT R9,0 (R9 serait mis à jour avec 10 sur C0=11), cela ne changerait rien pour C9 et C4, qui repasseraient à 0 sur la ligne suivante.

Ainsi sur la 3ème ligne, on aura C9=R9=0 et C4=R4=0.

3ème ligne :



Cette ligne ne pose pas davantage de problème, car la condition de dernière ligne est de nouveau fausse sur C0=6 (HSYNC) et vraie sur C0=7 (Hors HSYNC) (bien qu'elle ait été vraie sur C0=0, 1 et 2), ce qui désarme l'incrémentation de C4 et arme sa remise à 0. Et ainsi de suite…

Remarque : Pour des raisons pratiques il est parfaitement possible de déplacer l'instruction OUT R9,1 de 4 µsec (sur C0=4) (R9 est différent de C9 sur C0=R2+R3-1).

Enfin, tout comme sur le CRTC 0 pour « sortir » d'un traitement de « **dernière ligne** », il faut garder à l'esprit que lorsqu'on remet R9 à une valeur supérieure à 0, cette valeur n'est plus prise en compte si le CRTC considère qu'il est sur une dernière ligne.

Il va remettre C9 et C4 à 0 sur la ligne suivante, tel que le ferait le CRTC 0.

Si on a laissé la condition de dernière ligne active durant la dernière HSYNC (il y a eu deux dernières lignes avec un même C9/R9), C9 sera traité en relation avec la valeur de R9. C9 sera incrémenté si C9<>R9 comme le ferait un CRTC 1, mais C4 sera cependant incrémenté inconditionnellement si C9=R9 (et C9 repassera à 0).

Dans la perspective d'un code commun avec le CRTC 0, il est conseillé de :

- Traiter la mise à jour de R9 sur la dernière ligne comme sur les lignes précédentes pour que la prochaine ligne soit considérée comme une dernière ligne, afin d'être compatible avec le fonctionnement « dernière ligne » du CRTC 0.
- Ne pas modifier R9 avant C0=2 pour les mêmes raisons.

12.5 CRTC 3, 4

Si R4 est mis à jour avec la valeur de C4 :

- Si on était sur la ligne C9 entre 0 et R9-1, alors C9=C9+1, C4 passera à 0 lorsque C9 repassera à 0 et l'offset (R12/R13) est pris en compte.
- Si on était sur la dernière ligne (C9=R9), alors C9 passe à 0, C4=0 et R12/13 sont pris en compte.

La modification du registre 4 est prise en compte immédiatement à la fin de la ligne.
Si on veut mettre R4 à 0 pour que C4 boucle à 0, il faut donc le faire quand C4=0.

Si R4 est mis à jour avec une valeur inférieure à C4, alors il y a débordement du compteur C4. (contrairement à ce qui se produit avec C9/R9).

A noter également qu'une gestion additionnelle de lignes verticale n'incrémente pas C4 au-delà de R4, comme le font les CRTC 0, 1 et 2.

Créer une rupture ligne à ligne sur ce CRTC est encore plus trivial que sur CRTC 1, car il suffit de mettre R4 et R9 à 0 lorsque C4=0.

En effet, C9 passe à 0 si C9>R9, ce qui améliore le niveau de compatibilité entre les CRTC 0 et 1.

13 COMPTAGES : REGISTRE R0

13.1 GÉNÉRALITÉS

Le registre R0 du CRTC sert à définir le nombre de caractères que le circuit va générer sur une ligne. Un compteur C0 (aussi appelé HCC par quelques excentriques) compte de 0 jusqu'à la valeur de R0 incluse. Ce registre contient le nombre de caractères CRTC souhaité par « ligne » moins 1.

Lorsque C0 passe à 0, différents compteurs sont mis à jour (C4, « C5 », C9, ...).

Le compteur C0 est comparé à plusieurs autres registres, comme R1 (gestion du border/pointeur vidéo) et R2 (gestion HSYNC).

D'un point de vue algorithmique simple, on pourrait s'attendre à ce que la condition Cx=Rx définisse une incrémentation facultative d'autres compteurs.

Cependant, si C0 n'atteint pas certaines valeurs définies en interne, cela pose quelques problèmes, notamment sur le CRTC 0, car des traitements ont lieu sur des valeurs précises de C0.

Ces traitements spécifiques ont pour objet d'autoriser la remise à zéro de compteurs ou simplement gérer l’incrémentation d’autres compteurs.

En relation avec les timings décrits ci-après, et par rapport à ce qui suit, il est utile de rappeler qu'il y a un décalage entre les registres internes du CRTC et l'affichage effectif des caractères correspondants par le GATE ARRAY et/ou ASIC.

Le GATE ARRAY ne respecte pas ce "différé" pour générer visuellement les HSYNC. Les ASIC cependant, respectent ce différé pour afficher la HSYNC au moment ou le caractère correspondant (C0=R2) est affiché.

13.2 CRTC 0

Ce CRTC réalise des opérations (armements de remises à zéro ou incrémentations) de compteurs sur des positions précises de C0.

Le détail de ce fonctionnement permet de mieux appréhender le traitement spécifique de remise à 0 de C4, qui intervient seulement lorsque le CRTC évalue qu'il est sur la **dernière ligne de l'écran.**

Les concepteurs du circuit ont sans doute "préféré" gérer sous conditions une remise à 0 de C4, car ce compteur a été prévu pour **pouvoir dépasser R4** sur tous les CRTC « non ASICé ».

C'est notamment le cas lorsque R5 est supérieur à 0 (il devient alors la nouvelle limite de C9) ou que le mode Interlace est actif durant un écran « pair ».

Ces différents « tests » ont été répartis sur plusieurs valeurs de C0 :

- **Lorsque C0=0**, alors la condition « dernière ligne » C4=R4, C9=R9 est évaluée pour armer la mise à 0 de C4 (qui surviendra après la ligne additionnelle si il y a lieu).
- **Lorsque C0=1**, alors la gestion de C9 est autorisée pour le prochain C0=0. Si cette « autorisation » n'est pas donnée (contexte R0=0) alors le compteur C9 n'est plus géré sur le C0 suivant. Il faut toutefois noter que lorsque C9 est géré au passage de C0=0, il utilise la dernière valeur de R9 mise à jour avant que C0=0.
- **Lorsque C0=2**, et si la condition « dernière ligne » est vraie, alors le CRTC détermine si une gestion de ligne additionnelle sera activée pour le prochain C0=0 (C4 sera alors incrémenté, quelque soit la valeur de R4)

Ainsi :

- Si **R0=0**, alors C0 n'atteint jamais 1 (et reste à 0) et **C9 ne peut plus compter**.
  Il reste alors **figé sur la valeur qu'il avait avant que R0=0.**
- Si **R0=1**, alors C0 n'atteint jamais 2, et une gestion additionnelle de ligne(s) reste enclenchée si C4 valait R4 et C9 valait R9 sur le caractère précédent (C0=0).

Les conditions de remise à 0 de C4 sont multiples :

Lorsque C0=0, C4=R4 et C9=R9 conditionne l'état "**dernière ligne**".
Lorsque C0=2, si on est sur la "**dernière ligne**" et il n'y avait réellement aucune ligne additionnelle en attente (C9=R5+1 par exemple).

A défaut de remplir toutes ces conditions après C0>2, C4 sera incrémenté lorsque C0 repassera à 0. Autrement dit, **modifier R4 ou R9 après que les tests soient réalisés ne changera rien au résultat de C4**.

Si le CRTC n'était pas sur la dernière ligne (ou en ayant modifié R4 ou R9 avant que C0=0), le sort de C4 est déjà scellé : **C4=C4+1**.

GEL DE C9…
Le sort de C9 est particulier lorsque R0=0. En effet, ce dernier n'est plus incrémenté.
Au passage de R0=0, la valeur de C9 est calculée une première fois par rapport à R9.
Lorsque C9=R9, C4 est incrémenté une fois, mais C9 n'est pas remis à 0.

Si R4 valait 0 à la fin de la « ligne » pour laquelle R0 valait 0, alors C9 va déborder pour la nouvelle ligne. Le prochain C9 sera incrémenté **quelque soit la valeur de R9**.
Dès que C9 a commencé à déborder de cette manière, il va compter jusqu'à sa valeur limite quelque soient les valeurs qui pourront être programmées dans R9. Le CRTC est alors en gestion additionnelle, qui inhibe la remise à 0 de C9. C9 va alors être comparé à R5+1 et non plus R9.

Si R4 était différent de 0, alors la gestion C9=R9 est bien prise en compte. Il revient donc à 0, par exemple, lorsque C9=R9=0.

Remarque : Sur le CRTC 1, 2, 3 et 4, l'évaluation de « **dernière ligne** » a lieu durant toutes les valeurs de C0, **alors que le CRTC 0 ne le gère que sur C0=0**. Le CRTC 2 arme cependant une remise à 0 de dernière ligne qui ne peut plus être désarmée ensuite, quelles que soient les valeurs de R9 et R4. En respectant la plus forte contrainte, si R4 est modifié afin que sa valeur soit mise à jour sur C0=0 (OUT déclenché sur C0vs=#3E sur CRTC 0, 1, 2 et sur #3D sur CRTC 3, 4 pour R0=#3F) alors une seule instruction est nécessaire sans adaptation du code. Si toutefois il est aussi nécessaire de modifier R9, qui fait partie des conditions de « dernière ligne », il est nécessaire d'anticiper sa mise à jour.
Voir chapitre 12, page 54, et chapitre 10, page 42.

Il existe quelques subtilités complémentaires relatives à l'incrémentation de C4.

En particulier, si C4=R4 et C9=R9 lorsque C0=0.
Si une gestion d'ajustement vertical (lignes additionnelles) démarre, **C4 ne sera plus géré**.
(voir chapitre 11.2, page 50)

Elle peut démarrer lorsque C0 repasse à 0 si les conditions sont remplies, ou si le test de ces conditions n'a pas pu s'effectuer.

La gestion d'ajustement vertical a 2 particularités sur ce CRTC :

C4 est incrémenté une seule fois, quelque soit la valeur de R9 ou de R5.

C4 revient à 0 une fois l'ajustement terminé, quelque soit la valeur de R4.

(Cela signifie que C4 n'est plus comparé avec R4 pendant cette gestion spécifique de lignes additionnelles. Voir chapitre 11, page 49)

13.2.1 CAS PRATIQUE : R0=1

Lorsque R0 vaut 1 (et que C0<2) on obtient des "lignes" de 2 caractères (2 µsec).

Si C4=R4 et C9=R9 lorsque C0=0, alors on est potentiellement sur la dernière ligne de l'écran. Mais C0 n'ayant pas atteint 2, la gestion de ligne additionnelle n'est pas désarmée et va avoir lieu.

Si C4=R4 et C9=R9, alors "l'écran" prend fin lorsque C0 atteint R0.
C'est notamment le cas lorsque R4 et R9 valent 0, pour créer des "écrans" de 2 µsec.
(la première ligne est alors également la dernière)

Ainsi :

- Lorsque C0=0, le CRTC évalue si il est sur la dernière ligne, et si c'est le cas, arme un flag interne par défaut pour déclencher la gestion d'ajustement vertical.
- Lorsque C0=2, le CRTC évalue les conditions pour désarmer ce mécanisme d'ajustement vertical.

En conséquence :

- Si C0 ne dépasse jamais 1, et que R4=R9=0, le CRTC génère un ajustement vertical de "1 ligne" à chaque "écran" lorsque C4=R4 et C9=R9.
- Le CRTC ne peut plus désarmer la gestion additionnelle (test de R5 et/ou R8 (sous réserve)) et un "écran" de 2 µsec est généré. Cela se traduit par une incrémentation unique de C4 au-delà de la valeur programmée dans R4. L'ajustement cesse lorsque l'évaluation C5=R5 est vraie (C5 valant 0 au même titre que C9 sur la première ligne d'ajustement).
  Remarque 1 : C4 repasse à 0 une fois l'ajustement terminé.
  Remarque 2 : C9 repasse à 0 une fois l'ajustement terminé (sauf exception, voir R0=0).

Lorsque R4=R9=0, chaque "ligne" de 2 µsec est donc immédiatement suivie par une "ligne" de 2 µsec pour laquelle C9=0 et C4=1 (une exception existe cependant pour laquelle C9 n'est pas remis à 0, voir chapitre 0).

Une fois cette ligne "additionnelle" de 2 µsec terminée, C4 et C9 étant passés à 0, l'offset est pris en compte. Dans cette situation, les registres R12/R13 peuvent donc être pris en compte seulement après 4 µsec, alors que c'est possible toutes les µsec sur un CRTC 1, 2, 3 ou 4.

Si R9>0, cette "ligne" de 2µsec (pour laquelle C4=1) survient après la dernière valeur de C9 (lorsque C9=R9).

13.2.2 CAS PRATIQUE : R0=0

Lorsque R0 vaut 0 et que C0=0, alors C0 reste à 0 (écran ou ligne de 1 µsec).

Les choses se compliquent légèrement :

Blocage du comptage C9

Etant donné que C0 n'atteint jamais 1, C9 n'est plus géré.
Si par exemple, il valait 3 au moment ou R0 est passé à 0, il va rester à 3 quelque soit la valeur de R9, tant que R0=0.

Ligne additionnelle

Si la condition de dernière ligne est satisfaite (test en C0=0 de C4=R4 et C9=R9) alors la gestion de ligne additionnelle est armée :

- Elle ne sera désarmée que si C0 atteint 2 (et qu'aucune ligne additionnelle n'a été réellement programmée).
- Si cette gestion a lieu (C4 valait R4), alors C4 sera incrémenté une seule fois lorsque C0 repassera à 0, et reviendra à 0 ensuite. Dans cette situation cependant, on a vu que C9 reste figé. C4 a été incrémenté sans que C9 soit revenu à 0. Magique ! Tous les autres caractères seront alors égaux (C4=R4+1 et C9 figé) tant que R0=0.

Si la condition de dernière ligne n'était pas satisfaite (C4<>R4 ou C9<>R9) lorsque C0=0, alors la gestion de ligne additionnelle n'est pas armée :

- C4 ne pourra cependant plus s'incrémenter sur le prochain C0=0, car... C9 est figé.

La revanche de C9...

Lorsque la gestion additionnelle a lieu, C4 est incrémenté sans que C9 « évolue ». Se sentant abandonné, C9 jure alors de prendre sa revanche. Ce n'est sans doute pas la vraie raison, mais dans cette situation, **dès que C0 peut de nouveau dépasser 0, alors C9 est incrémenté quelque soit la valeur de R9 si R4 valait 0** lorsque C0 est de nouveau égal à 0.
La ligne additionnelle étant achevée, C4 passe alors à 0 sans que C9 soit remis à 0. La magie continue !
C9 ayant débordé à cause du déphasage de trigger, R9 n'a plus aucun effet sur le comparateur, et C9 va donc devoir revenir à 0 avant de pouvoir être évalué de nouveau avec R9.

Pour résumer :

C9 reste "figé" sur la valeur qu'il avait avant que R0=0 et ne s'incrémente donc plus, ni ne revient à 0.

Si une ligne additionnelle est ajoutée (C4=R4, C9=R9), alors
C4=R4+1 : C4 repassera à 0 une fois que C0 reviendra à 0.
C9 reste figé tant que R0=0.

Si R4 valait 0 et que R0 est de nouveau supérieur à 0 :
C9 sera incrémenté sans tenir compte de la valeur de R9 (C9=C9+1) dès que C0 atteindra de nouveau 0 et C4 repassera à 0. C9 ne sera géré de nouveau que lorsqu'il aura débordé.

Exemple 1:

Lorsque R0=0 et C0=0, et que C4=R4=C9=R9=0 :

1er caractère C0=0 | C9=0 | C4=0

Le pointeur vidéo courant est mis à jour (CRTC-VMA'=CRTC-VMA=R12/R13)
Lorsque C0 boucle une première fois sur R0, C0 passe de 0 à ... 0. C4=0.
Le CRTC est sur la fin "d'écran" (il vient de finir les lignes R4/R9 d'un écran de 1 µsec (C4=R4/C9=R9))

2ème caractère C0=0 | C9=0 | C4=1

Etant donné que C0 n'a pas atteint 2 et qu'on était sur une fin d'écran, la gestion de lignes additionnelles a lieu.
C4=C4+1 (soit 1). A noter que C9 n'est pas "remis à 0" car il n'est plus géré.

3ème caractère et suivants... C0=0 | C9=0 | C4=1

Le CRTC est en gestion additionnelle, et C4 ne peut plus s'incrémenter.
C4 n'évolue plus car il ne peut dépasser que de 1 (C4=1) et C9 reste figé.

Lorsque R0 redevient > 2 C0=0 | C9=0 | C4=1

Il faut attendre que C0=2 afin que la gestion de ligne additionnelle puisse cesser lorsque C0 repassera à 0.
C4 reste donc à 1 jusqu'à la fin de la ligne.
Cependant, à la fin de cette ligne "plus longue", C9 va s'incrémenter.

Ligne suivante, 1er caractère C0=0 | C9=1 | C4=0

C9 est passé à 1, et C4 est repassé à 0 (fin de gestion additionnelle)

Exemple 2: (suite du film)

On suppose ici que R0 est remis à 0 sur le 1er caractère de la fin de l'exemple 1.

1er caractère C0=0 | C9=1 | C4=0

Voir fin d'exemple 1 pour détail des valeurs.

2ème caractère et suivants... C0=0 | C9=1 | C4=0

C9 était différent de R9 sur le 1er caractère: pas de gestion additionnelle, C4=0 et C9 reste figé.

Ligne suivante, 1er caractère C0=0 | C9=2 | C4=0

C9 est en débordement à cause de la gestion additionnelle précédente, et il va compter jusqu'à revenir à 0.

13.2.3 CAS PRATIQUE : RUPTURE VERTICALE LAST LINE (R.V.L.L.)

UN PEU D'HISTOIRE TECHNIQUE... LA R.V.I.

Mise au point initialement sur le CRTC 0 à l'aide d'un canif et d'une ficelle, une ancienne technique ancestrale appelée "RVI" (Rupture Verticale Invisible © Overflow) a pour objet de permettre de choisir le numéro C9 de la ligne visible en créant de petites lignes dans la partie non visible de l'écran (durant la HSYNC), tout en modifiant l'adresse de la ligne visible. Pour y parvenir, il existe ***à minima*** *3 manières différentes, qui peuvent être combinées : Soit en modifiant R9, soit en modifiant la taille des lignes (R0), soit en limitant leur nombre (action synchrone Z80A/CRTC). Cela permet de pouvoir contrôler individuellement l'adresse de beaucoup plus de lignes que les 2k (parmi 16k d'une page) autorisés par la valeur C9=0.*

Une des contraintes majeure de cette technique est ***l'interdépendance entre les lignes.*** *Aussi vous apprécierez vraiment le prochain chapitre.*

En effet la valeur de C9 calculée pour la nouvelle ligne affichée dépend de la valeur de C9 de la ligne précédente. Créer un algorithme qui détermine automatiquement les évolutions souhaitées de C9 en fonction du C9 courant altère encore davantage la CPU disponible sur une ligne de 64 µsec.

C'est en principe la valeur de R9 qui fixe la valeur de C9 maximale. Il faut garder à l'esprit que seuls les 3 bits de C9 sont "conservés" pour la composition de l'adresse.

Un des corolaires de cette technique est en général la mise à jour de CRTC-VMA via une mise à jour de R12 et/ou R13. Pour cela il faut que C9 et C4 repassent à 0.
Remarque : Pour le CRTC 2, il faut de surcroit que R12/R13 soient modifiés avant que C0=R1 et que C0 atteigne R1 lorsque C9=R9, mais vous n'êtes pas sur le bon chapitre.

*Par exemple, si C9=8 correspond bien à C9=0 pour la constitution d'adresse, C4 ne repasse pas à 0 et donc R12/R13 ne peut pas être pris en compte. Pour qu'un changement d'adresse soit pris en compte, i****l faut que C9 "boucle" et repasse par 0, ainsi que C4****. Par ailleurs si le C9 affiché est une « dernière ligne », il va provoquer la remise à 0 de C9.*
Remarque : Pour le CRTC 1, il n'y a aucun besoin que C4 repasse à 0 pour que l'adresse soit prise en compte. C'est vrai à chaque fois que C0=0 tant que C4=0.

Lorsque des écran-lignes de 2 µsec sont créés dans le bord (ou pas si on veut exposer les dessous de l'affaire...) alors que R9>0, alors C9 va s'incrémenter jusqu'à atteindre R9.

Mais dans cette situation, lorsque C9=R9 (donc ici C0=1, puisque R0=1), une « ligne » additionnelle va être générée avec C4=1 et C9=0. Cette gestion "additionnelle" a pour conséquence la création d'une "ligne" complémentaire lorsque C9 atteint R9.

Le changement d'adresse n'a pas lieu car C4=1. C'est sur l'écran ligne suivant que C4 repassera à 0 (avec C9). Une des conséquences est que le nombre d'incrémentations de C9 pour ces "lignes de 2µs" est réduit de 1 par rapport aux CRTC 1, 2, 3, 4.

Cela ne permet pas d'accéder, avec R0=1, à C9=7 ou 5 en changeant l'offset, car il faut d'une part que ***R9 soit égal à 7 ou 5, mais qu'il ait pu reboucler à 0 avant****.*

Les valeurs finales de C9 impaires n'arrangent pas les choses, sauf si vous êtes un pro de l'Interlace "controlé" (bla, bla, avocat, bla, bla), mais je m'égare...

Cette technique ayant été développée sur le CRTC 0, l'usage de R0=0 a été abandonné rapidement à cause de la difficulté à appréhender le comportement des compteurs à l'époque.

Sur les CRTC 1, 2, 3 et 4, cette valeur (R0=0) ne pose aucun problème. Ni blocage de C9, ni ligne additionnelle C4=1, mais c'est plus subtil pour le CRTC 2. Il est donc bien plus simple de créer ce type de ruptures dans le bord permettant d'accéder à toutes les valeurs de C9 sur la ligne principale. La remise à 0 « armée » de « dernière ligne » est ici particulièrement intéressante par rapport au CRTC 1.

Au premier abord, la notion de « **dernière ligne** » sur le CRTC 0 (et le 2) parait contraignante, car elle ne permet pas une prise en compte « immédiate » des mises à jour des registres R4 et R9.

Par ailleurs, le CRTC 0 est limité à faire des ruptures de 2 µsec minimum pour un fonctionnement « correct » du compteur C9. Enfin, si une « dernière ligne » se produit lorsque cette ligne fait 2 µsec, alors une nouvelle ligne de 2 µsec est générée (C4=1).

Cependant, le principe de « **dernière ligne** » représente un formidable outil pour **programmer à l'avance une remise à 0 de C4 et C9**, notamment dans le cadre d'une rupture verticale, que je vais nommer Rupture Verticale Last Line (en bon franglais qui se respecte).

En effet, si on considère que chaque ligne visible est une « dernière ligne », alors **chaque première rupture de 2 µsec sera la première ligne d'un écran avec C4=C9=0**.
Ainsi il est possible d'anticiper la modification de R9 en sachant que le prochain C9 sera égal à 0.

Dans cette situation, la sélection du prochain C9 est grandement simplifiée puisqu'il suffit de mettre le numéro souhaité dans R9 et de créer le nombre de ruptures approprié (durant la HSYNC si l'objectif est de les cacher) afin que C9 sur la ligne suivante soit égal au R9 programmé.

La mise à jour de l'offset (R12/R13) a alors lieu lors de la première rupture de 2 µsec.
Etant donné que la dernière ligne (C9=R9) est toujours la ligne visible dont la taille est supérieure à 2 µsec, le CRTC ne génère pas de rupture de 2 µsec complémentaire.

Cela signifie qu'il faut 7 ruptures de 2 µsec pour atteindre C9=7, soient 14 µsec au total.

Cela signifie aussi, sauf à tricher de manière barbare avec la valeur des offset, que les ruptures de 2 µsec commencent plus ou moins tôt dans les 14 µsec afin d'amener C9 à la valeur souhaitée.

A cette fin, la valeur de R0 doit donc correspondre au moment ou les ruptures de 2 µsec commencent. Dis autrement R0 doit être égal à 63 – (R9 x 2) (pour R9>1).

Sur le schéma ci-dessous les zones en rose correspondent aux instructions OUT sur R0. R0-1 correspond à la mise à jour de R0 au début de la ligne (et donc débutée sur la fin de la ligne précédente), et R0-2 correspond à la valeur de R0 pour les ruptures « cachées ».

R0-1	R0-2	R9
63	x	0
50	12	1
59	1	2
57	1	3
55	1	4
53	1	5
51	1	6
49	1	7

			R2	OUT R0-2												OUT R0-1	New C9
C0:	... 47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	0
C0:	... 47	48	49	50	**0**	1	2	3	4	5	6	7	8	9	10	11	12	1
C0:	... 47	48	49	50	51	52	53	54	55	56	57	58	59	**0**	1	0	1	2
C0:	... 47	48	49	50	51	52	53	54	55	56	57	**0**	1	0	1	0	1	3
C0:	... 47	48	49	50	51	52	53	54	55	**0**	1	0	1	0	1	0	1	4
C0:	... 47	48	49	50	51	52	53	**0**	1	0	1	0	1	0	1	0	1	5
C0:	... 47	48	49	50	51	**0**	1	0	1	0	1	0	1	0	1	0	1	6
C0:	... 47	48	49	**0**	1	0	1	0	1	0	1	0	1	0	1	0	1	7
C3:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
																R3		

Remarque: La 2ème instruction OUT qui débute sur le 63ème caractère de la ligne a pour objet de redimensionner la ligne pour le C9 suivant. Cette instruction ne peut pas débuter sur le 64ème caractère sans provoquer un débordement de C4 sur la ligne redimensionnée (voir chapitre 13.6.2).

Sur un moniteur CTM, la partie visible d'une ligne est de 48 µsec, et 16 µsec ne sont pas visibles. Cependant, pour que l'écran soit centré, il faut que la HSYNC débute sur le 51ème caractère, lorsque C0 atteint 50 (si R0=63). Et donc que R2 soit programmé avec cette valeur.

Pour disposer des 14 µsec nécessaires pour atteindre C9=7, il faut que R0 soit programmé avec 49 et il faudrait en principe que R2 soit à 0 sur cette ligne précise (en inhibant toutefois les HSYNC en protégeant la zone avec une valeur de R3 couvrant toutes les autres occurrences de C0=0).

Une HSYNC n'étant pas forcément nécessaire sur chaque ligne, il est toutefois possible de positionner R2 avec 50, en sachant que les lignes C9=7 profitent de la synchronisation des autres lignes, mais cela implique qu'il n'y ait pas plusieurs lignes C9=7 consécutives.
C'est ce qui est fait dans la démonstration R.V. fournie avec SHAKER, ou les lignes 7 ne sont pas synchronisées. Cela ne pose aucun problème sur des moniteurs natifs.

Pour pouvoir utiliser toute la ram sans cette contrainte, il suffit de positionner R2=49.
Si C9=7 n'est pas utilisé, il est possible de positionner R2=50.

13.3 CRTC 1

R0 accepte toutes les valeurs sans que cela pose de problème aux autres compteurs.

Si R0 vaut 0, alors C9 et R4 continuent d'être gérés normalement.

L'offset est modifiable selon les timings indiqués dans les schémas des pages suivantes.

Remarque : Il est possible de créer 14 ruptures "cachées", permettant d'accéder simplement à toutes les valeurs de C9, et ce d'autant que l'offset est pris en compte tant que C4=0 (et non pas lorsqu'il repasse à 0).

Remarque : La mise à jour d'un des registres d'offset prenant à minima 4 µsec, la seule instruction Z80A de mise à jour couvre 4 "écrans lignes C0" lorsque R0=0.

Remarque : L'utilisation de l'instruction OUTI pour modifier R0 lorsque C4=C9=0 a des conséquences inattendues par rapport à cette même modification réalisée avec un OUT(C),R8. La modification du registre R0 est effective sur la 5ème µseconde de l'instruction OUTI, et sur le 3ème µseconde de l'instruction OUT(C),R8. En conséquence, l'instruction OUT(C),R8 doit débuter 2 µseconde plus tard que l'instruction OUTI. Mais dans cette circonstance, la HSYNC ou le BORDER sont affectés. Ce bug ayant été reproduit sur 2 machines différentes avec un CRTC 1 natif.

Je n'ai pas encore défini complètement ce qui se produit dans cette situation (dans la prochaine version du document), mais cela n'implique ni la position de la HSYNC (R2), ni la taille de la ligne (R1), et cela n'a lieu que lorsqu'un nouvel « écran » débute sur C0=0 lorsque C4=C9=0 (Sur C0>0 il n'y a pas de problème notable). La valeur de R0 a cependant une importance sur le bug et selon la machine. J'ai constaté une détérioration de la situation dans le temps sur un de mes CPC.

13.3.1 CAS PRATIQUE : RUPTURE VERTICALE INVISIBLE (R.V.I.)

La R.V.I., sur un CRTC 0, est très limitative car placer R0 < 2 pose quelques contraintes et cela limite le nombre de ruptures qu'il est possible de faire dans la partie « invisible » de la ligne.
En effet, lorsque R0 vaut 1, une ligne additionnelle est générée, et lorsque R0 vaut 0, C9 est affecté. Cette technique implique que C9 repasse à 0 pour que R12 et/ou R13 soi(en)t pris en compte, et cela nécessite impérativement un nombre élevé de ruptures.
Elle est impossible sur le CRTC 2, notamment à cause des contraintes de prise en compte de R12/R13.

Lorsqu'il est question d'écrire un programme fonctionnant sur les autres CRTC que le 0, la valeur R0=0 ne pose aucun problème particulier. Si l'objectif reste d'accéder à tous les C9, on préfèrera néanmoins la R.V.L.L. sur les CRTC 0 et 2 qui permet de s'affranchir d'un algorithme de gestion transitionnelle entre les C9. Les CRTC 1, 3 et 4 ne disposent pas de cette possibilité.

Le CRTC 1, par rapport aux CRTC 3 et 4 (et même si le CRTC 3 a bien d'autres moyens pour modifier l'offset de chaque ligne) dispose d'un avantage complémentaire pour gérer une R.V.I (si je ne l'ai pas déjà écrit). En effet, l'offset est pris en compte tant que C4=0, quelque soit la valeur de C9, contrairement à tous les autres CRTC. (voir chapitre 20.3).

Cela permet donc de modifier l'offset sans que C9 ait « rebouclé » à 0, et permet de limiter le nombre de ruptures nécessaires pour atteindre le C9 souhaité avec un offset « neuf », ce qui permet de placer R2 à 50 pour un centrage parfait.

Les tableaux suivants décrivent les 64 états transitionnels entre 2 C9. La méthode employée implique une mise à jour de R0 1 ou 2 fois par ligne, ainsi qu'une mise à jour de R9, R12 et/ou R13. Une joie pour tous les adeptes des SudokuZ80A !

Les valeurs des nouveaux C9 (indiqués en rouge) correspondent à une possibilité « C4=0 » du CRTC 1. Pour les CRTC 3 et 4, il suffit de remplacer les valeurs par celles indiquées sous chaque tableau. On notera que le passage C9=0 à 7 implique que R0 de la ligne principale soit égal à 49, limitant de facto la possibilité de mettre R2 à 50. Ce n'est pas du tout gênant pour un CRTC 4, puisque c'est la valeur qui permet le centrage sur ce circuit. (voir chapitre 16.1)

C9=0

Old C9	R0.1	R0.2	R9
0	63		0
0	63		1
0	59	0	2
0	57	0	3
0	55	0	4
0	59	0	5
0	58	0	6
0	57	0	7

	R2															OUT R02				OUT R01		New C9
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63				0
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63				1
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0				2
C0:	…	47	48	49	50	51	52	53	54	55	56	57	0	0	0	0	0	0				3
C0:	…	47	48	49	50	51	52	53	54	55	0	0	0	0	0	0	0	0				4
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0				5
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	0	0	0	0	0				6
C0:	…	47	48	49	50	51	52	53	54	55	56	57	0	0	0	0	0	0				7

C3:	1	2	3	4	5	6
						R3

Compatibility CRTC 3 & 4

Old C9	R0.1	R0.2	R9
0	59	0	1
0	53	0	5
0	51	0	6
0	49	0	7

			R2																		New C9
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0			1
C0:	…	47	48	49	50	51	52	53	0	0	0	0	0	0	0	0	0	0			5
C0:	…	47	48	49	50	51	0	0	0	0	0	0	0	0	0	0	0	0			6
C0:	…	47	48	49	0	0	0	0	0	0	0	0	0	0	0	0	0	0			7

C9=1

Old C9	R0.1	R0.2	R9
1	63		1
1	59	0	4
1	63		2
1	58	0	3
1	56	0	4
1	54	0	5
1	59	0	6
1	58	0	7

	R2															OUT R02				OUT R01		New C9
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63				0
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0				1
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63				2
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	0	0	0	0	0				3
C0:	…	47	48	49	50	51	52	53	54	55	56	0	0	0	0	0	0	0				4
C0:	…	47	48	49	50	51	52	53	54	0	0	0	0	0	0	0	0	0				5
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0				6
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	0	0	0	0	0				7

C3:	1	2	3	4	5	6
						R3

Compatibility CRTC 3 & 4

Old C9	R0.1	R0.2	R9
1	59	0	3
1	52	0	6
1	49	0	7

				R2																	New C9
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0			2
C0:	…	47	48	49	50	51	52	0	0	0	0	0	0	0	0	0	0	0			6
C0:	…	47	48	49	50	0	0	0	0	0	0	0	0	0	0	0	0	0			7

C9=2

Old C9	R0.1	R0.2	R9
2	63		2
2	59	0	2
2	58	0	2
2	63		3
2	57	0	4
2	55	0	5
2	53	0	6
2	59	0	7

	R2															OUT R02				OUT R01		New C9
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63				0
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0				1
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	0	0	0	0	0				2
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63				3
C0:	…	47	48	49	50	51	52	53	54	55	56	57	0	0	0	0	0	0				4
C0:	…	47	48	49	50	51	52	53	54	55	0	0	0	0	0	0	0	0				5
C0:	…	47	48	49	50	51	52	53	0	0	0	0	0	0	0	0	0	0				6
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0				7

C3:	1	2	3	4	5	6
						R3

Compatibility CRTC 3 & 4

Old C9	R0.1	R0.2	R9
2	59	0	3
2	51	0	7

				R2																	New C9
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0			3
C0:	…	47	48	49	50	51	0	0	0	0	0	0	0	0	0	0	0	0			7

C9=3

Old C9	R0.1	R0.2	R9
3	63		3
3	58	0	3
3	57	0	3
3	56	0	3
3	63		4
3	56	0	5
3	54	0	6
3	52	0	7

		R2											OUT R02					OUT R01		New C9
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	0
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	0	0	0	0	0	1
C0:	…	47	48	49	50	51	52	53	54	55	56	57	0	0	0	0	0	0	2
C0:	…	47	48	49	50	51	52	53	54	55	56	0	0	0	0	0	0	0	3
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	4
C0:	…	47	48	49	50	51	52	53	54	55	56	0	0	0	0	0	0	0	5
C0:	…	47	48	49	50	51	52	53	54	0	0	0	0	0	0	0	0	0	6
C0:	…	47	48	49	50	51	52	0	0	0	0	0	0	0	0	0	0	0	7
C3:						1	2	3	4	5	6								
R3																			

Compatibility CRTC 3 & 4

Old C9	R0.1	R0.2	R9
3	58	0	4

		R2																	New C9
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	0	0	0	0	0	4

C9=4

Old C9	R0.1	R0.2	R9
4	63		4
4	57	0	4
4	56	0	4
4	55	0	4
4	59	0	4
4	63		5
4	55	0	6
4	53	0	7

		R2											OUT R02					OUT R01		New C9
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	0
C0:	…	47	48	49	50	51	52	53	54	55	56	57	0	0	0	0	0	0	1
C0:	…	47	48	49	50	51	52	53	54	55	56	0	0	0	0	0	0	0	2
C0:	…	47	48	49	50	51	52	53	54	55	0	0	0	0	0	0	0	0	3
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0	4
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	5
C0:	…	47	48	49	50	51	52	53	54	55	0	0	0	0	0	0	0	0	6
C0:	…	47	48	49	50	51	52	53	0	0	0	0	0	0	0	0	0	0	7
C3:						1	2	3	4	5	6								
R3																			

Compatibility CRTC 3 & 4

Old C9	R0.1	R0.2	R9
4	57	0	5

		R2																	New C9
C0:	…	47	48	49	50	51	52	53	54	55	56	57	0	0	0	0	0	0	5

C9=5

Old C9	R0.1	R0.2	R9
5	63		5
5	59	0	8
5	59	0	7
5	59	0	6
5	59	0	5
5	58	0	5
5	63		6
5	54	0	7

		R2											OUT R02					OUT R01		New C9
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	0
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0	1
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0	2
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0	3
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0	4
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	0	0	0	0	0	5
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	6
C0:	…	47	48	49	50	51	52	53	54	0	0	0	0	0	0	0	0	0	7
C3:						1	2	3	4	5	6								
R3																			

Compatibility CRTC 3 & 4

Old C9	R0.1	R0.2	R9
5	56	0	6

		R2																	New C9
C0:	…	47	48	49	50	51	52	53	54	55	56	0	0	0	0	0	0	0	6

C9=6

Old C9	R0.1	R0.2	R9
6	63		6
6	59	0	9
6	59	0	8
6	59	0	7
6	59	0	6
6	58	0	6
6	57	0	6
6	63		7

					R2								OUT R02					OUT R01	New C9
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	0
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0	1
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0	2
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0	3
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0	4
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	0	0	0	0	0	5
C0:	…	47	48	49	50	51	52	53	54	55	56	57	0	0	0	0	0	0	6
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	7
					C3:	1	2	3	4	5	6								
											R3								

Compatibility CRTC 3 & 4

Old C9	R0.1	R0.2	R9
6	55	0	7

					R2														New C9
C0:	…	47	48	49	50	51	52	53	54	55	0	0	0	0	0	0	0	0	7

C9=7

Old C9	R0.1	R0.2	R9
7	63		7
7	59	0	10
7	59	0	9
7	59	0	8
7	59	0	7
7	58	0	7
7	57	0	7
7	56	0	7

					R2								OUT R02					OUT R01	New C9
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	0
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0	1
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0	2
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0	3
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	0	0	0	0	4
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	0	0	0	0	0	5
C0:	…	47	48	49	50	51	52	53	54	55	56	57	0	0	0	0	0	0	6
C0:	…	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	7
					C3:	1	2	3	4	5	6								
											R3								

13.4 CRTC 2

R0 accepte toutes les valeurs sans que cela pose de problème aux autres compteurs.

Cependant, une des particularités de ce CRTC est d'inhiber beaucoup de traitements durant la HSYNC. Voir chapitres 15 (page 92) et 14 (page 84).

En particulier :

- La condition VSYNC qui se produit lorsque C4=R7 est annulée pour le reste du frame si cela se produit durant la HSYNC. Voir chapitre 15, page 92.
- La gestion de remise à 0 de C4 (lorsqu'on est sur la dernière ligne d'un écran) est désactivée dans certaines conditions. Voir chapitres 12 (page 54) et 10 (page 42).
- La condition de désactivation du border (lorsque C0=0) n'est également plus gérée, et le BORDER de la ligne précédente reste actif. Voir chapitre 14, page 84.

Ce CRTC génère 1 octet (0.5µsec) de BORDER avant que C0=0, mais contrairement au CRTC 0, qui fait la même chose, la fonction R8 SKEW DISP n'existe pas pour contourner ce problème. On ne peut donc pas (à ma connaissance) empêcher l'apparition de 1 octet de BORDER.
Voir chapitre 19, page 123.

Le contenu de R12/R13 est "transféré" dans CRTC-VMA' lorsque C0 atteint R1.
Puis CRTC-VMA' est transféré dans CRTC-VMA en début d'écran.
Ce qui implique qu'il est impossible de changer d'offset si C0 n'atteint pas R1.
Voir chapitre 14, page 84.

Changer l'offset implique donc qu'au moins une microseconde de BORDER existe avant que C0=R0, afin que C0 puisse être égal à R1.

Le traitement de C4 et C9 sur ce CRTC suit en partie la logique du CRTC 0, dans la mesure où il existe une notion d'armement de la remise à 0 ou de l'incrémentation de C4 sur la condition de « **dernière ligne** ». Le CRTC détermine si il est sur une dernière ligne sur les différentes positions de C0, et va armer la remise à 0 de C9 (et de C4) pour la prochaine ligne, et ce quelques soient les valeurs programmées après dans R9 et R4 durant la ligne. Cette notion à une importance capitale pour permettre la gestion d'une RVLL sur ce CRTC.

Lorsqu'il est nécessaire de créer les conditions permettant un changement d'offset à chaque « ligne », C4, R4, C9, R9 doivent tous être à 0 lorsque C0=0 et C0 doit avoir rencontré R1 après que R12 et/ou R13 soient mis à jour.

Sans une petite feinte de sioux, C4 va déborder sur la deuxième ligne.
Voir chapitre 10, page 42.

Si R0 est positionné à 3, par exemple, le CRTC 2 va générer 16 « lignes » de 4 µsec.
Dans cette situation, il faut veiller à ce que R2 soit supérieur à 0 :

- Afin que la HSYNC débute obligatoirement lorsque C0>0 afin que la VSYNC soit prise en compte lorsque C4 atteint R7. Ceci peut cependant être « contourné ».
- Afin que le BORDER soit désactivé (ou du moins ne reste pas activé) lorsque C0=0, ce qui est une contrainte plus gênante.

13.4.1 CAS PRATIQUE : RUPTURE VERTICALE LAST LINE (R.V.L.L.)

Le CRTC 2 collectionne un florilège de contraintes diverses qu'il est nécessaire de prendre en compte si on souhaite réaliser une rupture verticale...

Disons le tout de suite, effectuer une RVI « traditionnelle » sur ce CRTC est impossible pour une raison simple : Pour que l'offset soit pris en compte, il faut que C0 atteigne R1 **sur la dernière ligne**. **Modifier R12/R13 après que C0>R1 ne permet pas une prise en compte pour la ligne suivante**. Or, la RVI est basée sur le principe d'atteindre la dernière ligne dans le bord, et il faudrait donc qu'une micro-ligne « dans le bord » ait un C0 qui atteint R1 (lorsque C9=R9).

Mettre R1=1 lorsque R0=1 activera le BORDER à partir de C0=1.
Sauf à être capable de modifier R0 et R1 en même temps, plus grand chose ne sera affiché....

Il faut donc que C0 puisse atteindre R1 sur la ligne visible, qui doit également être la dernière de l'écran. **Cela tombe bien car la RVLL fonctionne sur ce principe !**

Le CRTC 2 n'est pas du tout gêné pour le comptage de C9 et C4 lorsque R0=0.
A ce titre il dispose du même potentiel de ruptures « cachées » qu'un CRTC 1, soit 14 ruptures de 1 µsec, ce qui aurait du permettre de régler le problème de centrage rencontré sur CRTC 0, dont le potentiel de ruptures cachées est de 7.
Mais, malheureusement, ceci est possible uniquement si 2 lignes avec C9 identique ne sont pas consécutives.

R0-1	R0-2	R9
63		0
58	4	1
57	2	2
57	1	3
55	1	4
58	0	5
57	0	6
56	0	7

				R2								OUT R02					OUT R01		New C9
C0:	...	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	0
C0:	...	47	48	49	50	51	52	53	54	55	56	57	58	0	1	2	3	4	1
C0:	...	47	48	49	50	51	52	53	54	55	56	57	0	1	2	0	1	2	2
C0:	...	47	48	49	50	51	52	53	54	55	56	57	0	1	0	1	0	1	3
C0:	...	47	48	49	50	51	52	53	54	55	0	1	0	1	0	1	0	1	4
C0:	...	47	48	49	50	51	52	53	54	55	56	57	58	0	0	0	0	0	5
C0:	...	47	48	49	50	51	52	53	54	55	56	57	0	0	0	0	0	0	6
C0:	...	47	48	49	50	51	52	53	54	55	56	0	0	0	0	0	0	0	7
C3:					1	2	3	4	5	6									
R3																			

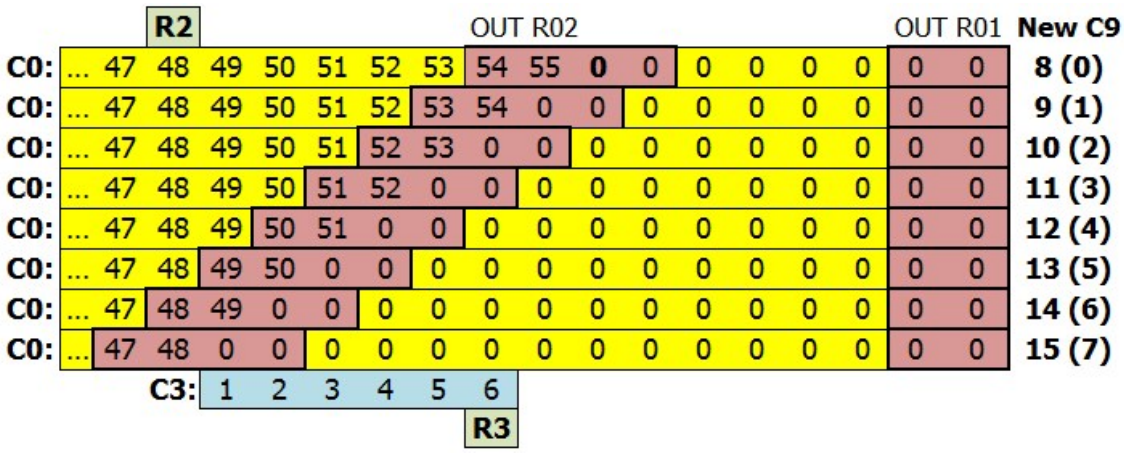
Tout comme le CRTC 0, il existe une notion de « **dernière ligne** » permettant de programmer une remise automatique à 0 de C9 et C4. Cependant, cette notion n'est pas complète car cette remise à 0 automatique est désactivée si 2 HSYNC consécutives se produisent avec le même C9=R9.

Autrement dit, 2 lignes avec le même C9=R9 durant 2 HSYNC consécutives annulent la remise à 0 de C4 (qui déborde alors salement). Il est impossible de modifier R9 durant la HSYNC pour empêcher la HSYNC d'annuler cette remise à 0 (comme on peut le faire dans une rupture ligne à ligne) car si on souhaite modifier R9 durant la HSYNC, aucun truc magique ne permet de modifier R0 en parallèle pour créer les ruptures cachées.

Cependant, il est fort utile de rappeler ici que R9 est un registre 5 bits dont seuls 3 bits participent au pointeur vidéo. Lorsque la ligne C9=8 est affichée, on visualise la ligne C9=0.

Il est donc possible de gérer les contraintes de contiguïté des C9 en utilisant ce principe, comme on peut le voir sur le tableau suivant :

R0-1	R0-2	R9
55	0	8
54	0	9
53	0	10
52	0	11
51	0	12
50	0	13
49	0	14
48	0	15



La démonstration fournie avec SHAKER utilise un écran sans C9 identiques contigus utilisant les 8 blocs. La première et la dernière ligne utilisent cependant ce principe « C9 and 7 » afin d'éviter cette contiguïté (en créant une ligne 0 via C9=8 et une ligne 1 via C9=9).

Il convient toutefois de noter que pour atteindre C9=6 et C9=7 (via C9=14 et 15), il est nécessaire que R0 atteigne 49 et 48 respectivement. Ce n'est pas une situation idéale si les lignes visibles doivent être centrées.

Plus C0=R2 se produit tôt, plus les « ruptures cachées » apparaissent à gauche sur l'écran.

Les plus curieux pourront toujours aller jeter un œil dans l'intro de « AMAZING DEMO 2021 ».
Une méthode simple pour contourner la contrainte des lignes contigües consiste à alterner 1 ligne sur 2 la table de gestion des C9 à atteindre. Ainsi une ligne 1 est alors forcément suivie d'une ligne C9+8 et le problème est ainsi réglé.

Remarques : Dans la mesure où le CRTC 0 ne peut pas créer des ruptures cachées de 1 µsec, cela signifie que les premières données à apparaitre à gauche sont le 3ème octet de la ligne, suivi d'une ½ µsec de BORDER. Sur un CRTC 2, ce serait le 1er octet de la ligne (lorsque R0=0) suivi d'une ½ µsec de BORDER. Enfin sur les CRTC 1, 3, 4 dans le même contexte R2, ce serait 2 octets qui apparaitraient à gauche.

13.5 CRTC 3, 4

R0 accepte toutes les valeurs sans que cela pose de problème aux autres compteurs.

Si R0 vaut 0, alors C9 et R4 continuent d'être gérés normalement.

L'offset est modifiable selon les timings indiqués dans les schémas des pages suivantes, chapitre 13.7.

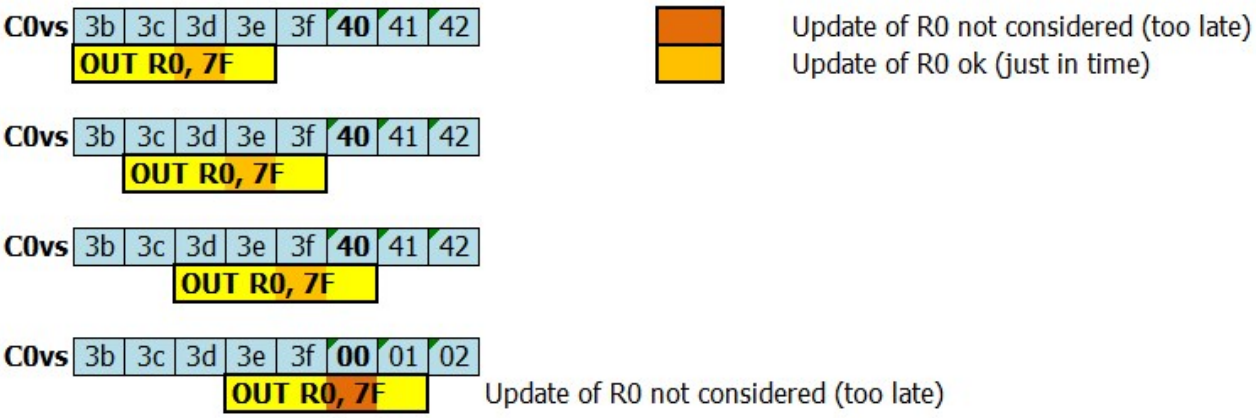
Les techniques de R.V.L.L. sont inapplicables sur ces CRTC. La principale « subtilité » de ces CRTC étant le passage de C9 à 0 si R9 est programmé est avec une valeur inférieure au C9 courant (qui permet une « certaine » compatibilité avec le CRTC 0). Il est possible de réaliser une R.V.I. avec la contrainte d'un bouclage de C9 à 0 pour que l'offset soit pris en compte. (voir chapitre 13.3.1). Attention néanmoins à ne pas oublier que l'I/O a lieu avec 1 µs de retard, ce qui implique de placer les OUT 1 µsec avant ceux qu'on ferait sur un CRTC 1.

13.6 MISE A JOUR DE R0

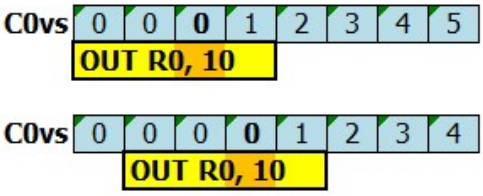
13.6.1 DÉLAIS DE PRISE EN COMPTE

CRTC 0, 1, 2

Previous R0=#3f

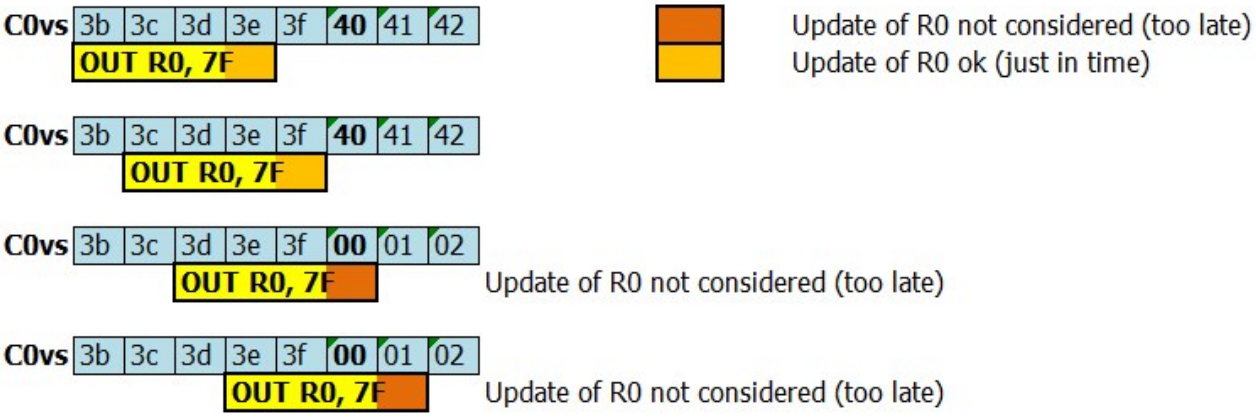


Previous R0=0

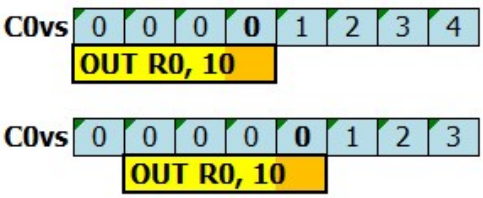


CRTC 3, 4

Previous R0=#3f



Previous R0=0



La mise à jour du registre R0 est prise en compte immédiatement par le CRTC pour le comptage de C0. Le compteur C0 ne dépasse jamais la valeur de R0 lorsque R0 est mis à jour selon le timing décrit sur les schémas précédents.
Autrement dit, si R0=0 et que C0 vaut 0, il ne débordera pas, et C0 vaudra toujours 0.

En général, la valeur programmée dans R0 est 63 afin que 64 µsec s'écoulent avant que le compteur repasse à 0. Ce bouclage permet au compteur C0 de repasser sur la valeur de R2 dans une période de 64 µsec.

Lorsque R0 est reprogrammé durant le balayage d'une ligne, il faut prévoir de reprogrammer R2 pour que la HSYNC ait lieu au même endroit toutes les 64 µsec et qu'il n'y en ait qu'une seule de plus de 2 µsec.

En principe, il est plutôt conseillé de générer 1 HSYNC par ligne, mais un moniteur CTM supportera de ne pas en avoir 1 par ligne (bla, bla, hérétiques, bla, bla…, puristes, bla, bla,…).

Comme indiqué précédemment, sur le CRTC 0, la mise à jour de R0 avec une valeur inférieure à 2 empêche le CRTC de réaliser certains traitements spécifiques qui ont des conséquences sur le comptage.

Le délai minimum possible entre l'affection de R13 et R12 (dans cet ordre) est de 8 µsec.
Voir chapitre 20, page 149.

13.6.2 EXCEPTIONS

Il existe très certainement d'autres exceptions qui restent encore à établir dans une jolie table mais voici néanmoins quelques cas identifiés, selon les CRTC.

CRTC 1

Il existe une différence de prise en compte par le CRTC 1 de la modification de R0 sur une position précise de C0 selon l'instruction Z80A utilisée pour cette mise à jour. Les deux instructions concernées sont OUTI (I/O sur la 5ème µsec) ou OUT(C),reg8 (I/O sur la 3ème µsec). Ceci traduit une différence de délai de traitement interne au niveau des instructions Z80A.

C'est le cas lors de la mise à jour de R0 sur la position C0=0 lorsque C4=C9=0. Une RVI avec des ruptures de 1 NOP sur ce CRTC ne peut pas utiliser OUTI. Voir chapitre 11, page 49. Je n'ai pas (encore) identifié exactement ce qui se produit, mais cela semble avoir une action sur la désactivation du BORDER, et non sur le comptage de C0, n'ayant pas constaté de désynchronisation dans cette situation.

CRTC 0

Lorsque C0=R0, le CRTC va incrémenter C4 si une remise à 0 n'a pas été programmée.
Cette remise a 0 est totalement effective partir de C0=2 lorsque le CRTC a évalué qu'il est sur une dernière ligne écran (C4=C4 et C9=R9) et qu'il n'y a pas de gestion additionnelle (C9=R5, R8=1 ou 3 sur le premier frame ou le registre est mis à jour).

Pour bien comprendre ce que je vais décrire, il faut avoir en tête que lorsque la condition C0=R0 se produit après C0=2 dans le cadre d'une gestion additionnelle, alors C4 est incrémenté quelque soit la valeur de R4, et C9 passe à 0 lorsque C0 revient à 0. A priori (à défaut de la preuve du contraire), le CRTC 0 fait l'économie de C5 et utilise C9 lors de la gestion additionnelle.

Si R0 est programmé avec la valeur 1, alors le test de gestion additionnelle n'est pas réalisé, et l'incrémentation de C4 reste programmée lorsque C9=R9. Dans le cadre des délais de mise à jour de R0, cette gestion peut être partiellement accomplie si la mise à jour à lieu sur C0=1 alors que R0=1 et qu'on veut le modifier avec une valeur supérieure à 1 (par exemple pour refaire passer une ligne à 64 µsec...).

Si la mise à jour de R0 à lieu lorsque **C0=1 et C9=R9**, cela provoque un débordement de C4 sans que C0 et C9 soient remis à 0, **même si C4 n'est pas égal à R4**.
On est en gestion de ligne additionnelle :

- C4 est incrémenté inconditionnellement
- C9 ne peut plus repasser à 0, même lorsque C9=R9.

Le test d'évaluation C0=R0 (avec la valeur 1) a lieu avant que le registre soit mis à jour, ce qui provoque le « début d'une gestion additionnelle ». Mais le registre R0 a cependant été mis à jour assez tôt pour être pris en compte pour l'incrémentation de C0 tel qu'indiqué dans le chapitre précédent (voir chapitre 13.6.1) :

	Screen Grid:	56	57	58	59	60	61	62	63	0	1	2	3	4	5	6	7
	C0:	...	56	57	58	59	**0**	1	0	1	0	**1**	2	3	4	5	6	7
R9=7	C9:	...	4	4	4	4	5	5	6	6	7	7	7	7	7	7	7	7
R4>=6	C4:	...	6	6	6	6	6	6	6	6	6	6	7	7	7	7	7	7

OUT R0,1 OUT R0,63

Dans cet exemple, deux situations sont possibles sur la ligne C9=7 lorsque R9=7 et lorsque C0 atteint R0 (qui vaut alors 63 grâce au second OUT).

13.6.2.1 On n'était pas sur une « dernière ligne écran» (C4<>R4, C9=R9)

La gestion additionnelle est désactivée d'office sur la position C0=2 (C4<>R4) car R5 vaut 0.
La conséquence est alors juste un débordement de C4 à partir de C0=2, sans que C4 repasse à 0 ensuite comme il l'aurait fait en gestion additionnelle.
A noter que **ce débordement n'aurait pas eu lieu si C9 avait été différent de R9.**
Dans l'exemple, C4 serait resté égal à 6.
Le contrôle du comptage C9/C4 reste classique.
Si, par exemple, il était nécessaire de compenser les deux « lignes » de 2 µsec créées après C0=59 afin que 8 lignes complètes soient affichées, il faudrait programmer R9 avec 9 pour ajouter 2 nouvelles lignes de 64 µsec.

13.6.2.2 On était sur une « dernière ligne écran » (C4=R4, C9=R9)

L'état dernière ligne a été activé avec C4=R4, C9=R9, quelque soit la valeur de R4.
C'est vrai également si R4=0.
Comme dans le cas précédent, C4 est incrémenté à partir de C0=2.
La gestion additionnelle est évaluée sur la position C0=2.
Mais C9 vaut alors 7 car il n'a pas été remis à 0 dans cette situation.
C9 étant alors différent de R5, la gestion additionnelle est activée.
Si, par exemple, il était nécessaire de compenser les deux « lignes » de 2 µsec créées après C0=59 afin que 8 lignes complètes aient été affichées, **il faudrait programmer R5 avec 10** pour ajouter 2 nouvelles lignes 64 µsec. Si R5 n'est pas reprogrammé et valait 0, C9 va s'incrémenter pour afficher les lignes 8 à 31, jusqu'à ce qu'il atteigne R5.
Lorsque la gestion additionnelle est terminée, alors C4 repasse à 0.

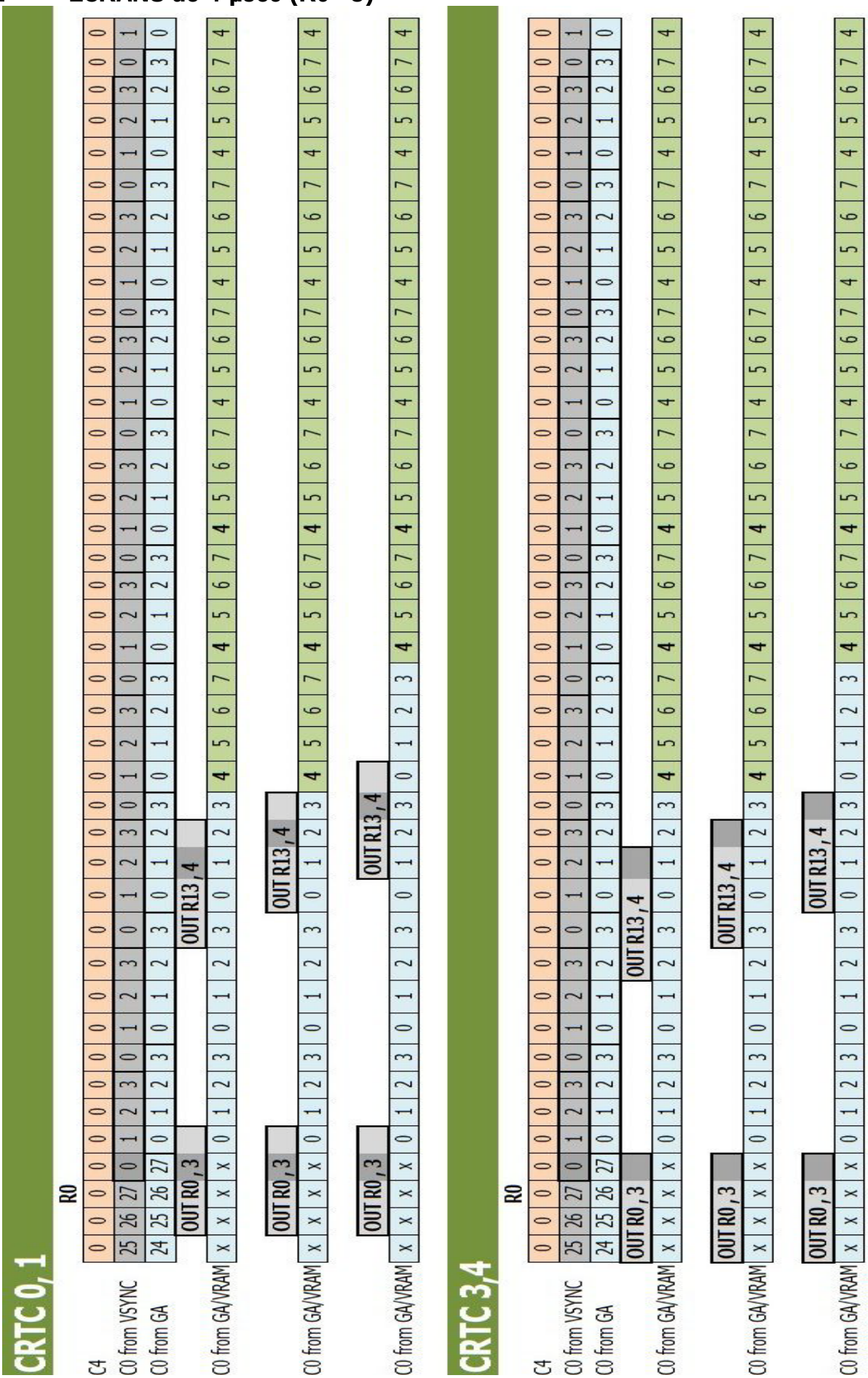
Si on veut éviter que C4 soit incrémenté lors d'un « agrandissement » de R0 qui valait 1, il faut donc le faire impérativement sur la position C0=0.

13.7 OFFSET SELON C0

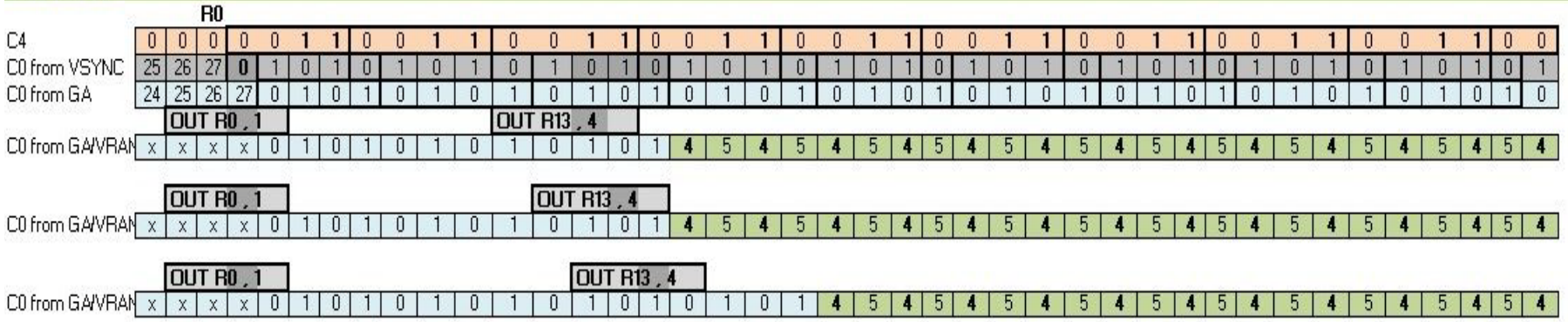
Les schémas suivants décrivent la prise en compte d'un changement d'offset (R12 et/ou R13) par rapport à la valeur courante de **C0vs**. Le CRTC 2 n'est pas représenté ici car ces tests ont été réalisés initialement avec une valeur de R1 supérieure à R0.

Les changements d'offset sont représentés avec la couleur verte.
Données initiales :
CRTC-R4=0
CRTC-R9=0
CRTC-R1=4
CRTC-R13=0

13.7.1 ECRANS de 4 µsec (R0=3)

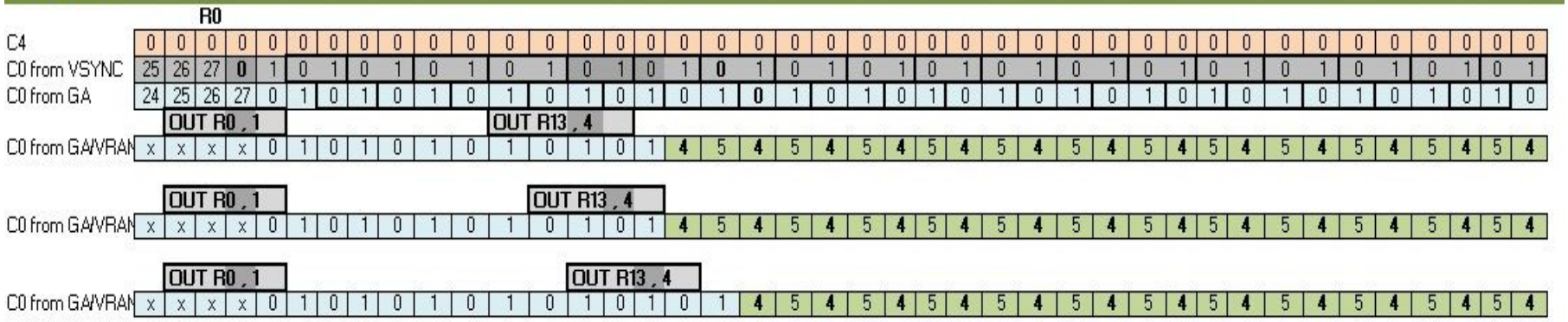


CRTC 0

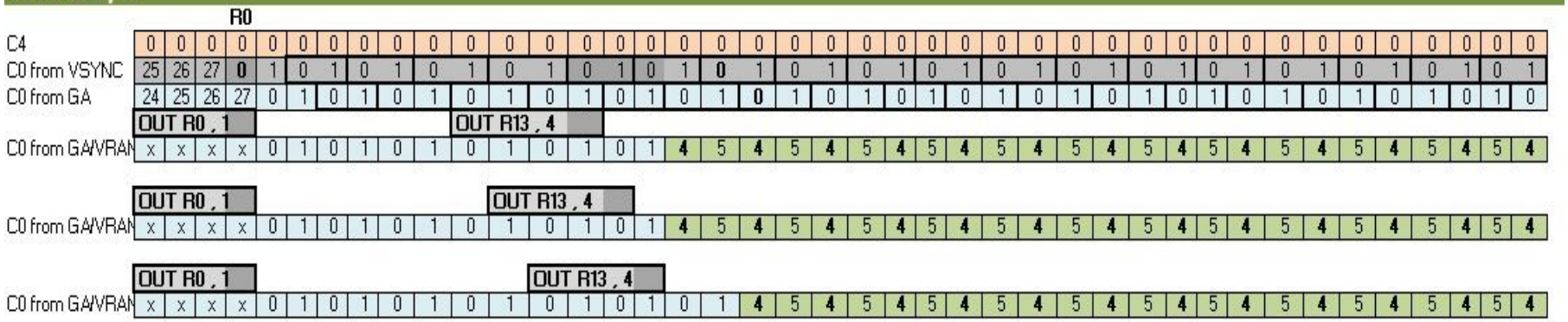


L'évènement C0=R0 au bout de 2 µsec laisse C4=1 pendant la 2ème période de 2 µsec, qui représente un ajustement vertical "non annulé" (car C0 n'est jamais égal à 2)

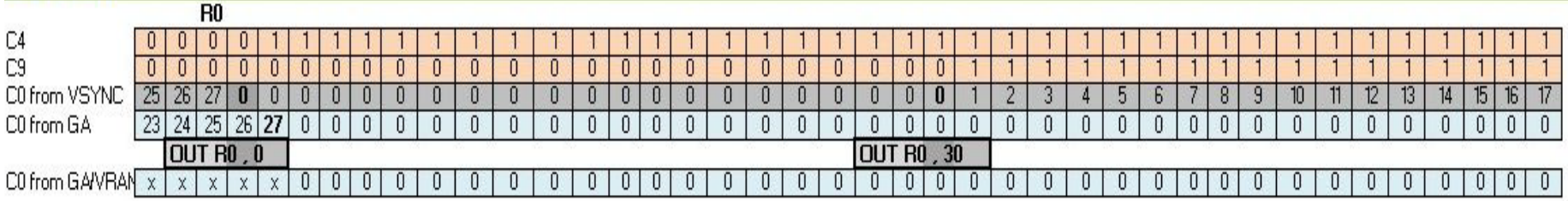
CRTC 1



CRTC 3, 4

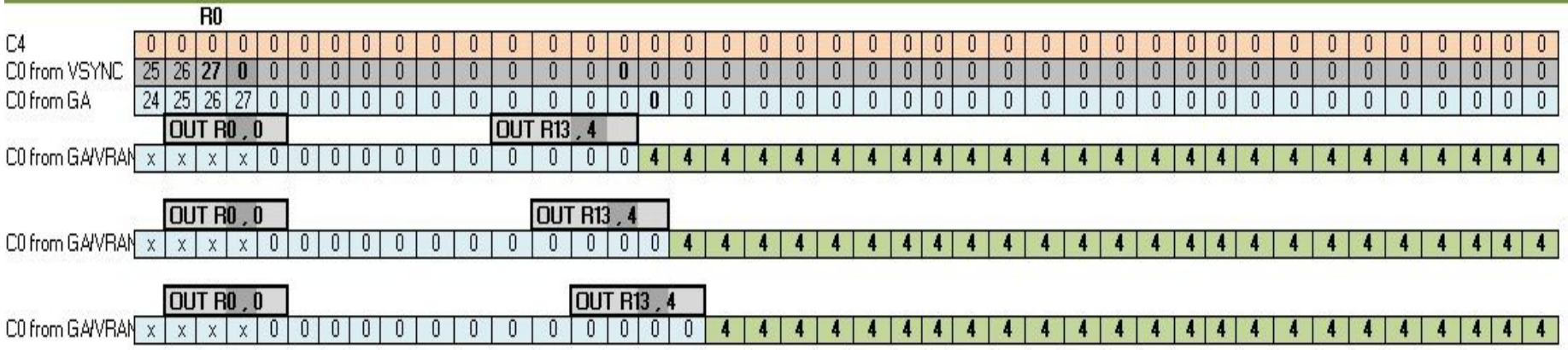


CRTC 0

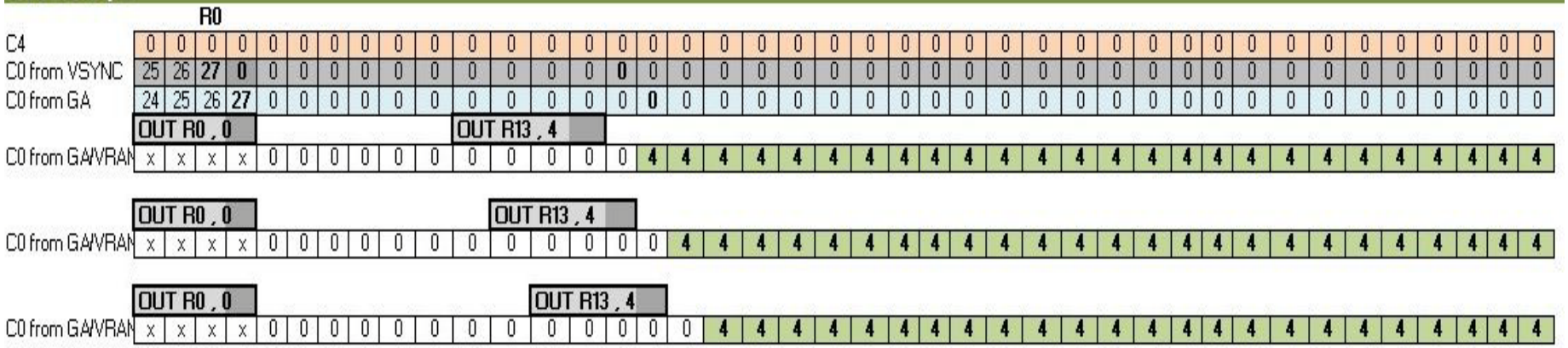


La prise en compte de R12/R13 ne peut pas avoir lieu tant que C4 et C9 ne repassent pas tous les deux à 0

CRTC 1



CRTC 3,4



14 SYNCHROS : REGISTRE R3

14.1 GÉNÉRALITÉS

Le registre 3 peut contenir 2 informations différentes, selon le modèle du CRTC 6845.

De manière générale, il permet de fixer :

- La durée de la HSYNC.
- La durée de la VSYNC pour certains CRTC.

On peut considérer un compteur C3 qui passe à 1 sur C0=R2 et qui compte jusqu'à atteindre la valeur de R3, ce qui correspond le plus à mes observations.

CRTC	7	6	5	4	3	2	1	0
0	Vsync	Vsync	Vsync	Vsync	Hsync	Hsync	Hsync	Hsync
1	x	x	x	x	Hsync	Hsync	Hsync	Hsync
2	x	x	x	x	Hsync	Hsync	Hsync	Hsync
3	Vsync	Vsync	Vsync	Vsync	Hsync	Hsync	Hsync	Hsync
4	Vsync	Vsync	Vsync	Vsync	Hsync	Hsync	Hsync	Hsync

Autres CRT	7	6	5	4	3	2	1	0
MC6845*1	Vsync	Vsync	Vsync	Vsync	Hsync	Hsync	Hsync	Hsync
UM6845E	Vsync	Vsync	Vsync	Vsync	Hsync	Hsync	Hsync	Hsync

Hsync CRTC				
0	0	0	0	**Pas de Hsync**
0	0	0	1	1 nop
0	0	1	0	2 nop
0	0	1	1	3 nop
0	1	0	0	4 nop
0	1	0	1	5 nop
0	1	1	0	6 nop
0	1	1	1	7 nop
1	0	0	0	8 nop
1	0	0	1	9 nop
1	0	1	0	10 nop
1	0	1	1	11 nop
1	1	0	0	12 nop
1	1	0	1	13 nop
1	1	1	0	14 nop
1	1	1	1	15 nop

Hsync CRTC				
0	0	0	0	**16 nop**
0	0	0	1	1 nop
0	0	1	0	2 nop
0	0	1	1	3 nop
0	1	0	0	4 nop
0	1	0	1	5 nop
0	1	1	0	6 nop
0	1	1	1	7 nop
1	0	0	0	8 nop
1	0	0	1	9 nop
1	0	1	0	10 nop
1	0	1	1	11 nop
1	1	0	0	12 nop
1	1	0	1	13 nop
1	1	1	0	14 nop
1	1	1	1	15 nop

Vsync CRTC				
0	0	0	0	**16 lignes**
0	0	0	1	1 ligne
0	0	1	0	2 lignes
0	0	1	1	3 lignes
0	1	0	0	4 lignes
0	1	0	1	5 lignes
0	1	1	0	6 lignes
0	1	1	1	7 lignes
1	0	0	0	8 lignes
1	0	0	1	9 lignes
1	0	1	0	10 lignes
1	0	1	1	11 lignes
1	1	0	0	12 lignes
1	1	0	1	13 lignes
1	1	1	0	14 lignes
1	1	1	1	15 lignes

14.2 LONGUEUR VSYNC

Les CRTC de première génération génèrent une VSYNC de 16 lignes.

Les constructeurs ont fait évoluer le circuit en ajoutant une fonction permettant de paramétrer le nombre de lignes, en utilisant les 4 bits de poids fort de R3.

Afin d'assurer une compatibilité des programmes créés pour la première génération de circuits, la valeur 0 pour les CRTC intégrant la nouvelle fonction correspond à 16 lignes de VSYNC.

L'intégralité des 4 bits de poids fort de R3 sert à indiquer un nombre de lignes exact, sauf pour 0 donc, qui signifie 16 lignes.

La ROM du CPC initialise VSYNC à 8 (R3=1000xxxx).

Ainsi, les CRTC 1 et 2, qui ne gèrent pas ces bits, génèrent 16 lignes de VSYNC alors que les CRTC 0, 3 et 4 en génèrent 8. (lorsque R7 est déjà programmé avant que C4=R7).

Etant donné que les CRTC 1 et 2 ne savent pas gérer une VSYNC paramétrable, **il est conseillé d'éviter d'utiliser cette fonction** sur les CRTC 0, 3 et 4 si l'objectif est la compatibilité.

Cela implique de reprogrammer ce registre car l'initialisation par la rom crée une différence qui pourrait être évitée.

Cela peut entrainer des incompatibilités pour un programme qui aurait la bonne idée de se synchroniser sur la fin de la VSYNC par exemple.

14.3 HSYNC GATE ARRAY VERSUS CRTC

La durée maximale de la HSYNC traitée par le GATE ARRAY pour la synchronisation de l'image est de 6 µsec, plafonnée par la durée programmée via la HSYNC CRTC.

Dis autrement si la HSYNC CRTC est supérieure à 6, cela n'a aucune incidence sur la synchronisation horizontale du moniteur.

Le GATE ARRAY produit un signal de synchronisation composite utilisé par le moniteur.
Les signaux HSYNC et VSYNC produits par le CRTC sont fusionnés en 1 seul signal de synchronisation, qui est présent sur la broche 4 du connecteur vidéo DIN6.

Cependant, il peut être utile d'avoir une HSYNC « CRTC » supérieure à 6 car le CRTC ne prend pas en compte la mise à jour de tous ses registres de la même manière durant cette période et cela permet donc d'inhiber des traitements.

14.4 HSYNC ET POSITION ECRAN

Il est possible d'utiliser la taille de la HSYNC-GA pour décaler l'écran de environ 1/2 caractère CRTC (~ 1 octet). Ce "mécanisme" étant analogique, le résultat dépend du réglage du moniteur CTM et de sa tolérance.

Il peut produire des variations de couleur ou des résultats imprévus sur d'autres écrans et même créer l'apparition impromptue d'acronymes de modes graphiques.

Les valeurs utilisées en général, sans trop affecter la synchronisation, sont 5/6 ou 4/5.

Une valeur de 5 provoque un décalage de l'image à droite par rapport à une valeur de 6 (valeur maximale), quelque soit la valeur de R2.

Dans la même logique, une valeur de 4 provoque un décalage de l'image à droite par rapport à une valeur de 5.

Remarque : Sur le jeu "*Skatewars*", Jon MENZIES utilise les valeurs &85 ,&8E, ce qui revient aux valeurs 5/6 pour la HSYNC-GA.

14.5 HSYNC ET INTERRUPTIONS

La modification de la taille de la HSYNC modifie le moment ou une interruption est générée.
Le GATE ARRAY arme une interruption juste après la fin de la HSYNC, sous certaines conditions.
Voir chapitre 25, page 163

14.6 MISE A JOUR DE R3 DURANT LA HSYNC

Il est possible de modifier la valeur de R3 lorsque C3 compte, ce qui peut avoir une incidence sur la longueur de la HSYNC.

Si R3 est modifié avec une valeur inférieure à C3, alors C3 est en débordement, sauf pour le CRTC 1 avec la valeur 0, qui annule la HSYNC en cours.

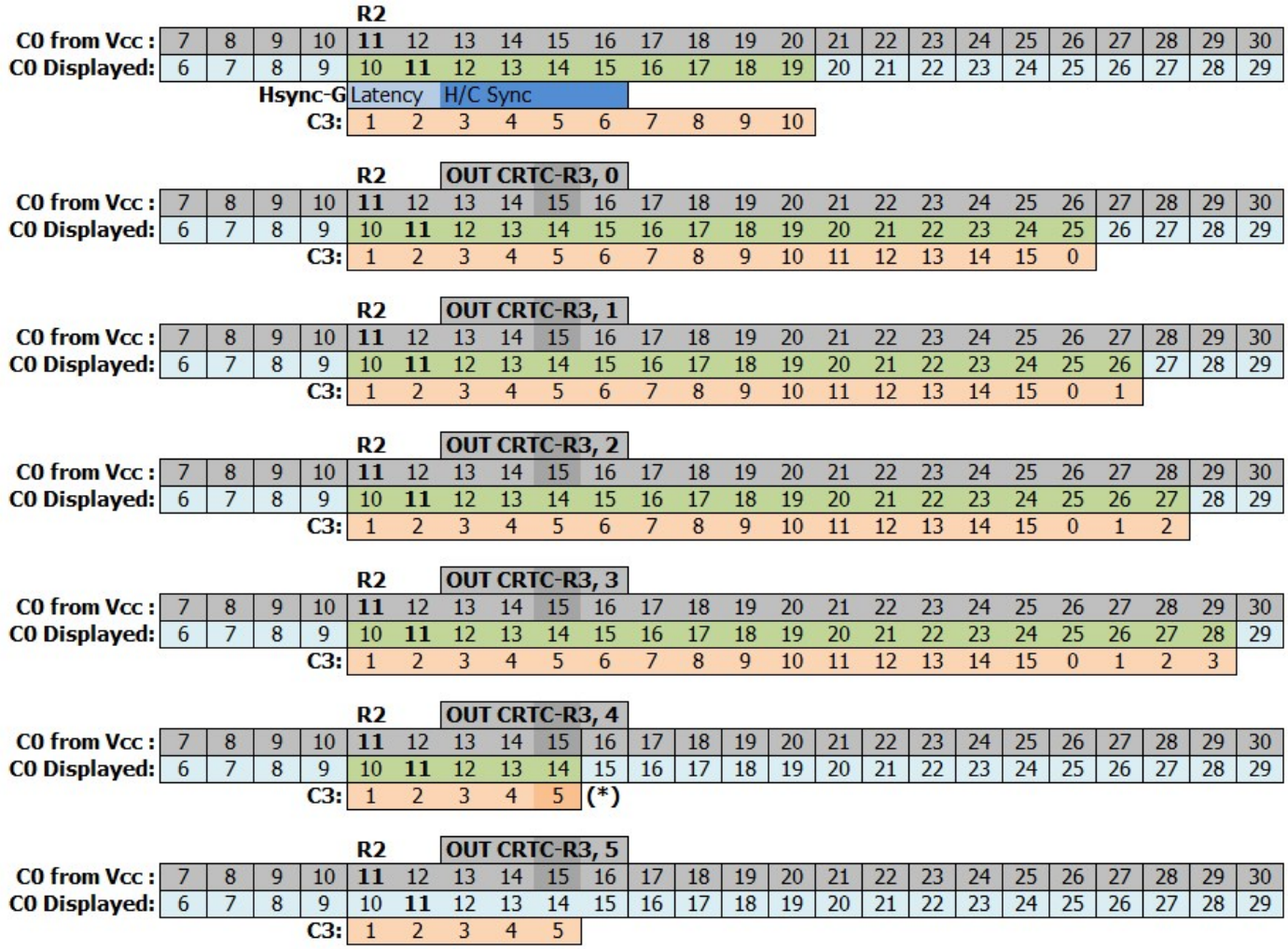
La mise à jour de R3 durant ce décompte est prise en compte selon les situations décrites ci-après

Légende pour les schémas pages suivantes :

	Z80A (OUT (C),r8) instruction
	Register update
	HSYNC Zone
	Characters displayed

14.6.1 CRTC 0, 2

CRTC-R2=11 / CRTC-R3=10 (HBL Size = 10 chars)



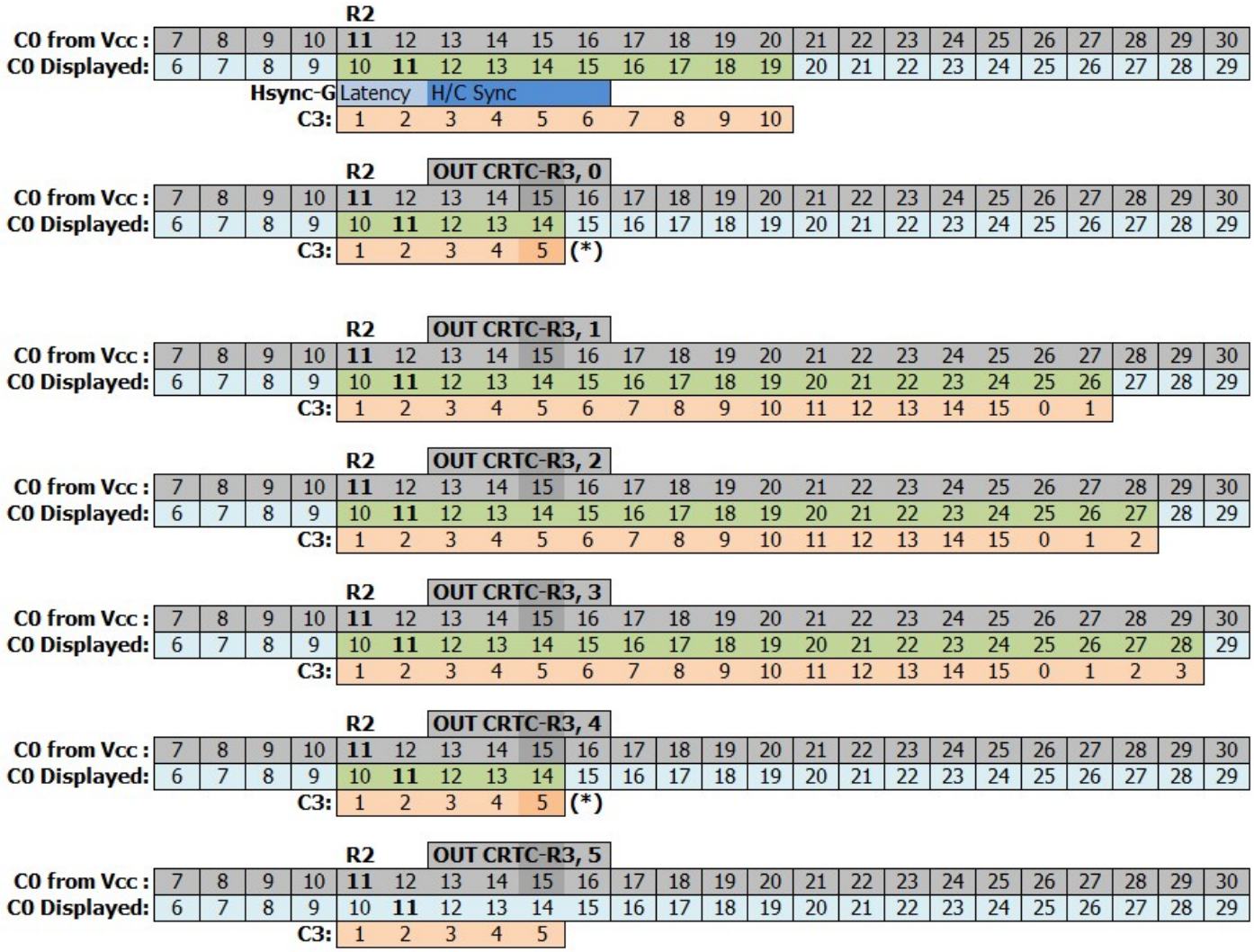
(*) Comme pour R2 qui est retardé par le fonctionnement propre du Z80A, la mise à jour précise de R3 interrompt prématurément le « non affichage » de la HSYNC après 0.5 µsec (1/2 caractère pour C0vs=15).

14.6.2 CRTC 1

CRTC-R2=11 / CRTC-R3=10 (HBL Size = 10 chars)

Durant une mise à jour de R3 pendant la HSYNC le CRTC 1 se comporte comme les CRTC 0 et 2, sauf lorsque R3 est mis à jour avec la valeur 0. En effet le traitement de R3=0 (absence de HSYNC) continue à être géré durant la HSYNC pour ce CRTC.

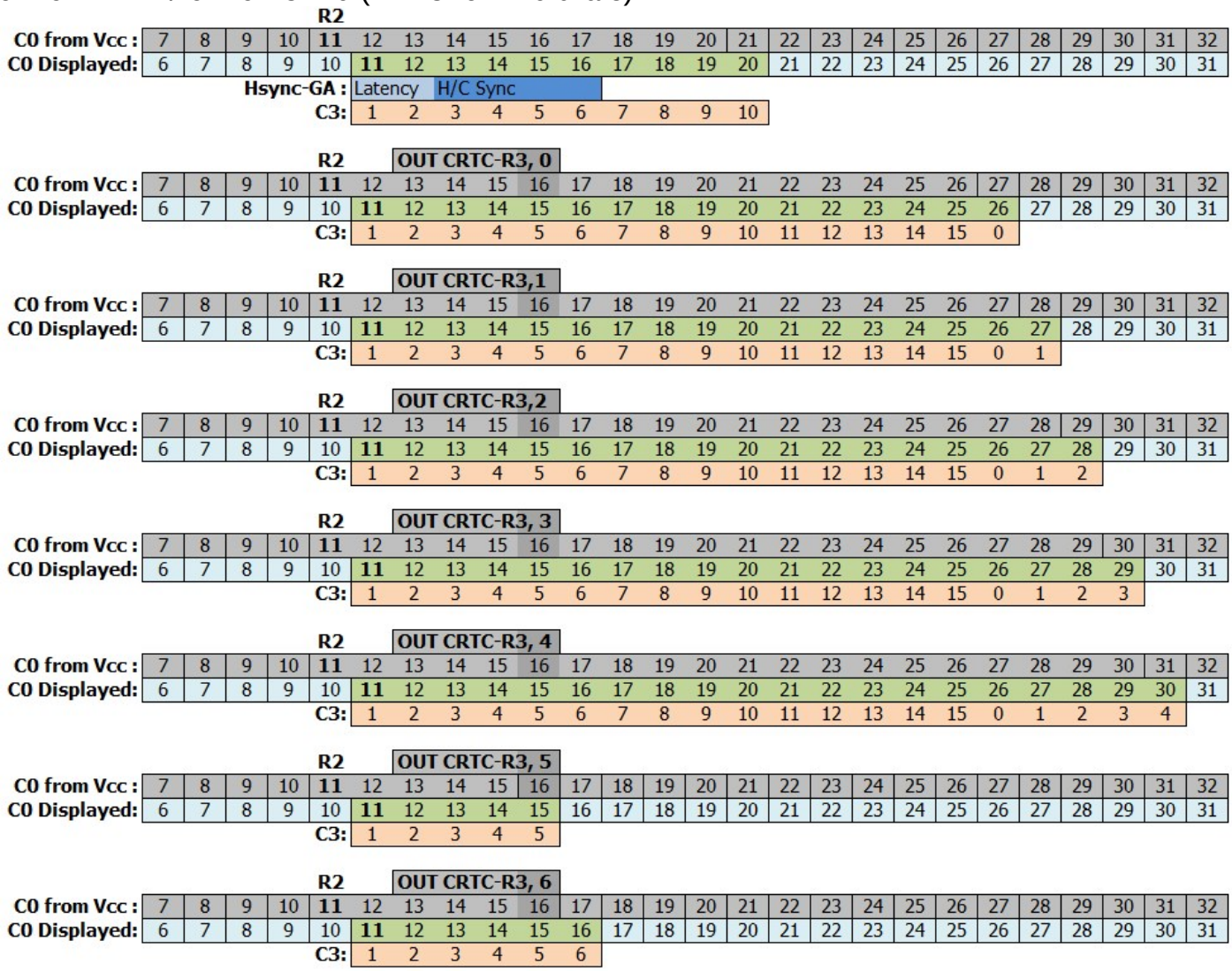
Sur les CRTC 0 et 2 dans cette condition, la HSYNC se poursuit et 0 est traité comme une valeur à atteindre.



(*) Comme pour R2 qui est retardé par le fonctionnement propre du Z80A, la mise à jour précise de R3 interrompt prématurément le « non-affichage » de la HSYNC après 0.5 µsec (1/2 caractère pour C0vs=15).

14.6.3 CRTC 3, 4

CRTC-R2=11 / CRTC-R3=10 (HBL Size = 10 chars)



14.7 ABSENCE DE HSYNC

Lorsque R3=0, les CRTC 0 et 1 ne produisent pas de HSYNC (**et donc pas d'interruption**).

Sur CRTC 2, 3 et 4, il est impossible de ne pas générer de HSYNC si la condition C0=R2 est remplie. Une valeur de 0 dans R3 va générer une HSYNC de 16 µsec, sauf si elle est interrompue en modifiant R3 pendant la HSYNC.

14.8 DEMARRAGE HSYNC

14.8.1 CRTC 0, 1, 2

Lorsque C0=R2 alors une HSYNC est générée sur une longueur de R3 caractères.
La mise à jour de R2 a lieu pendant la 3ème µsec de l'instruction OUT(C),reg8.

Lorsque la mise à jour de R2 a lieu avant que C0 atteigne R2, la HSYNC est traitée par le GATE ARRAY pendant l'affichage du caractère précédent, le caractère C0=R2 n'ayant pas encore été affiché par le GATE ARRAY.

Dans cette situation, la HSYNC "affichée" ne débute pas exactement sur une frontière de caractère. La HSYNC débute sur le 3ème pixel (en mode graphique 2) après le début du caractère CRTC R2-1 affiché (sous réserve).

Si cette mise à jour de R2 intervient pendant que C0=R2 (sur le 3ème µsec du OUT) alors le CRTC envoie le signal HSYNC plus tardivement au GATE ARRAY. Cela traduit un délai entre l'exécution de l'instruction en Z80A et la prise en compte par le CRTC.

Dans cette situation, la HSYNC "affichée" débute sur le 9ème pixel (en mode graphique 2) après le début du caractère CRTC R2-1 affiché.

La durée de la HSYNC reste décomptée avec la valeur programmée dans R3 (sauf si R3 est modifié durant la HSYNC).

Ce cas particulier de mise à jour permet de:

- Retarder le début de la HSYNC affichée de 6 pixels mode 2 (soit 0.4 µsec).
- Réduire la taille affichée de cette HSYNC de 0.4 µsec, ce qui peut s'avérer intéressant pour minimiser l'espace noir en cas de changement de mode graphique en cours de ligne. Il faut néanmoins tenir compte que le mode 2 est affiché 1 pixel (0.0625 µsec) plus tôt par le GATE ARRAY.

Sur le schéma page suivante, R3 est fixé à 2.
R2 avant modification est supérieur à 10.

14.8.2 CRTC 3, 4

Lorsque C0=R2 alors une HSYNC est générée.
Cette HSYNC est retardée comme l'affichage du caractère correspondant.
La mise à jour des registres CRTC étant également différée de manière synchronisée, la mise à jour ne peut plus dépendre de la vitesse à laquelle le OUT en Z80A est effectué.

Ainsi, si R2 est modifié après que C0 soit devenu égal à R2, alors R2 n'est pas pris en compte et la HSYNC n'est pas générée.

Les schémas ci-après illustrent les propos précédents en montrant le positionnement visuel de la HSYNC selon le moment ou la mise à jour de R2 a lieu.

CRTC 0, 1, 2

Z80a instr.		**OUT CRTC-R2, 11**				**I / O**									
C0 from Vcc		07		08		09		10		**11**		12		13	
C0 disp by Gate Array		06		07		08		09		10		11		12	
Byte Offset (0.5 µsec)		12	13	14	15	16	17	18	19	20	21	22	23	24	25
Pixel Mode 0		24 25	26 27	28 29	30 31	32 33	34 35	36 37	38 39	40 41	42 43	44 45	46 47	48 49	50 51
Pixel Mode 1		48 49 50 51	52 53 54 55	56 57 58 59	60 61 62 63	64 65 66 67	68 69 70 71	72 73 74 75	76 77 78 79	80 81 82 83	84 85 96 87	88 89 90 91	92 93 94 95	96 97 98 99	100 101 102 103
Pixel Mode 2		96 97 98 99 100 101 102 103	104 105 106 107 108 109 110 111	112 113 114 115 116 117 118 119	120 121 122 123 124 125 126 127	128 129 130 131 132 133 134 135	136 137 138 139 140 141 142 143	144 145 146 147 148 149 150 151	152 153 154 155 156 157 158 159	160 161 162 163 164 165 166 167	168 169 170 171 172 173 174 175	176 177 178 179 180 181 182 183	184 185 186 187 188 189 190 191	192 193 194 195 196 197 198 199	200 201 202 203 204 205 206 207

Z80a instr.		**OUT CRTC-R2, 10**				**I / O**									
C0 from Vcc		07		08		09		**10**		11		12		13	
C0 disp by Gate Array		06		07		08		09		10		11		12	
Byte Offset (0.5 µsec)		12	13	14	15	16	17	18	19	20	21	22	23	24	25
Pixel Mode 0		24 25	26 27	28 29	30 31	32 33	34 35	36 37	38 39	40 41	42 43	44 45	46 47	48 49	50 51
Pixel Mode 1		48 49 50 51	52 53 54 55	56 57 58 59	60 61 62 63	64 65 66 67	68 69 70 71	72 73 74 75	76 77 78 79	80 81 82 83	84 85 96 87	88 89 90 91	92 93 94 95	96 97 98 99	100 101 102 103
Pixel Mode 2		96 97 98 99 100 101 102 103	104 105 106 107 108 109 110 111	112 113 114 115 116 117 118 119	120 121 122 123 124 125 126 127	128 129 130 131 132 133 134 135	136 137 138 139 140 141 142 143	144 145 146 147 148 149 150 151	152 153 154 155 156 157 158 159	160 161 162 163 164 165 166 167	168 169 170 171 172 173 174 175	176 177 178 179 180 181 182 183	184 185 186 187 188 189 190 191	192 193 194 195 196 197 198 199	200 201 202 203 204 205 206 207

Z80a instr.				**OUT CRTC-R2, 10**				**I / O**							
C0 from Vcc		07		08		09		**10**		11		12		13	
C0 disp by Gate Array		06		07		08		09		10		11		12	
Byte Offset (0.5 µsec)		12	13	14	15	16	17	18	19	20	21	22	23	24	25
Pixel Mode 0		24 25	26 27	28 29	30 31	32 33	34 35	36 37	38 39	40 41	42 43	44 45	46 47	48 49	50 51
Pixel Mode 1		48 49 50 51	52 53 54 55	56 57 58 59	60 61 62 63	64 65 66 67	68 69 70 71	72 73 74 75	76 77 78 79	80 81 82 83	84 85 96 87	88 89 90 91	92 93 94 95	96 97 98 99	100 101 102 103
Pixel Mode 2		96 97 98 99 100 101 102 103	104 105 106 107 108 109 110 111	112 113 114 115 116 117 118 119	120 121 122 123 124 125 126 127	128 129 130 131 132 133 134 135	136 137 138 139 140 141 142 143	144 145 146 147 148 149 150 151	152 153 154 155 156 157 158 159	160 161 162 163 164 165 166 167	168 169 170 171 172 173 174 175	176 177 178 179 180 181 182 183	184 185 186 187 188 189 190 191	192 193 194 195 196 197 198 199	200 201 202 203 204 205 206 207

Z80a instr.		**OUT CRTC-R2, 9**				**I / O**									
C0 from Vcc		07		08		**09**		10		11		12		13	
C0 disp by Gate Array		06		07		08		09		10		11		12	
Byte Offset (0.5 µsec)		12	13	14	15	16	17	18	19	20	21	22	23	24	25
Pixel Mode 0		24 25	26 27	28 29	30 31	32 33	34 35	36 37	38 39	40 41	42 43	44 45	46 47	48 49	50 51
Pixel Mode 1		48 49 50 51	52 53 54 55	56 57 58 59	60 61 62 63	64 65 66 67	68 69 70 71	72 73 74 75	76 77 78 79	80 81 82 83	84 85 96 87	88 89 90 91	92 93 94 95	96 97 98 99	100 101 102 103
Pixel Mode 2		96 97 98 99 100 101 102 103	104 105 106 107 108 109 110 111	112 113 114 115 116 117 118 119	120 121 122 123 124 125 126 127	128 129 130 131 132 133 134 135	136 137 138 139 140 141 142 143	144 145 146 147 148 149 150 151	152 153 154 155 156 157 158 159	160 161 162 163 164 165 166 167	168 169 170 171 172 173 174 175	176 177 178 179 180 181 182 183	184 185 186 187 188 189 190 191	192 193 194 195 196 197 198 199	200 201 202 203 204 205 206 207

CRTC 3,4

Z80a instr.		**OUT CRTC-R2, 9**						**I / O**							
C0 from Vcc		07		08		09		**10**		11		12		13	
C0 disp by Gate Array		06		07		08		09		10		11		12	
Byte Offset (0.5 µsec)		12	13	14	15	16	17	18	19	20	21	22	23	24	25
Pixel Mode 0	**No Hsync**	24 25	26 27	28 29	30 31	32 33	34 35	36 37	38 39	40 41	42 43	44 45	46 47	48 49	50 51
Pixel Mode 1	**No Hsync**	48 49 50 51	52 53 54 55	56 57 58 59	60 61 62 63	64 65 66 67	68 69 70 71	72 73 74 75	76 77 78 79	80 81 82 83	84 85 96 87	88 89 90 91	92 93 94 95	96 97 98 99	100 101 102 103
Pixel Mode 2	**No Hsync**	96 97 98 99 100 101 102 103	104 105 106 107 108 109 110 111	112 113 114 115 116 117 118 119	120 121 122 123 124 125 126 127	128 129 130 131 132 133 134 135	136 137 138 139 140 141 142 143	144 145 146 147 148 149 150 151	152 153 154 155 156 157 158 159	160 161 162 163 164 165 166 167	168 169 170 171 172 173 174 175	176 177 178 179 180 181 182 183	184 185 186 187 188 189 190 191	192 193 194 195 196 197 198 199	200 201 202 203 204 205 206 207

Z80a instr.		**OUT CRTC-R2, 10**						**I / O**							
C0 from Vcc		07		08		09		**10**		11		12		13	
C0 disp by Gate Array		06		07		08		09		10		11		12	
Byte Offset (0.5 µsec)		12	13	14	15	16	17	18	19	20	21	22	23	24	25
Pixel Mode 0		24 25	26 27	28 29	30 31	32 33	34 35	36 37	38 39	40 41	42 43	44 45	46 47	48 49	50 51
Pixel Mode 1		48 49 50 51	52 53 54 55	56 57 58 59	60 61 62 63	64 65 66 67	68 69 70 71	72 73 74 75	76 77 78 79	80 81 82 83	84 85 96 87	88 89 90 91	92 93 94 95	96 97 98 99	100 101 102 103
Pixel Mode 2		96 97 98 99 100 101 102 103	104 105 106 107 108 109 110 111	112 113 114 115 116 117 118 119	120 121 122 123 124 125 126 127	128 129 130 131 132 133 134 135	136 137 138 139 140 141 142 143	144 145 146 147 148 149 150 151	152 153 154 155 156 157 158 159	160 161 162 163 164 165 166 167	168 169 170 171 172 173 174 175	176 177 178 179 180 181 182 183	184 185 186 187 188 189 190 191	192 193 194 195 196 197 198 199	200 201 202 203 204 205 206 207

15 SYNCHROS : REGISTRE R7

15.1 GÉNÉRALITÉS

Le registre R7 sert à fixer le moment ou le CRTC génère le signal VSYNC.

L'écran cale son affichage vertical à partir du moment où C4=R7.

C'est à partir de ce signal VSYNC que le canon à électron remonte en diagonale (droite vers gauche, bas vers haut) pour stabiliser l'image et continuer à afficher les caractères envoyés par le CRTC. Ces caractères ne sont pas affichés, mais le CRTC continue à gérer ses compteurs et pointeurs.

La taille de la VSYNC est exprimée en nombre de HSYNC.

Ce nombre de HSYNC est programmable sur CRTC 0, 3 et 4 (via le registre R3) et est fixé à 16 pour les CRTC 1 et 2.

Il est possible de générer plusieurs VSYNC durant un frame, mais le moniteur ne saura se caler que sur un signal VSYNC. Par contre, il est impossible de déclencher ou inhiber une VSYNC durant une VSYNC.

Lorsqu'une VSYNC est en cours, le GATE ARRAY n'affiche rien.
La zone est noire (à ne pas confondre avec la couleur d'un BORDER, qui est en soi un affichage).

Modifier R7 ne permet pas de désactiver la VSYNC une fois qu'elle a débuté.

Enfin, sur un moniteur CTM, l'image commence à être visible à partir de la 34ème ligne (Ce qui représente la deuxième ligne du 5ème caractère de 8 lignes à partir du début de la VSYNC).

15.2 CONDITIONS DE PRISE EN COMPTE

15.2.1 CRTC 0

La VSYNC débute lorsque C4=R7.

Si R7 est modifié avec la valeur de C4, alors la VSYNC est déclenchée immédiatement.
Si R7 est modifié sur C0vs=#36, par exemple, alors la prochaine lecture du PPI (au plus tôt 5 µsec après, donc sur C0=#3B) renvoie un statut actif de la VSYNC.

L'activation ainsi "déclenchée" de la VSYNC ne comptabilise pas la ligne.
Le compteur de lignes VSYNC est géré en début de ligne lorsque C0=0.

En conséquence, la durée totale de la VSYNC est augmentée du nombre de µsec correspondante au calcul R0 - C0vs (du moment où R7 a été mis à jour).

Si une VSYNC est déclenchée en cours de ligne numéro 1, alors la VSYNC se termine à la fin de la ligne 17.

15.2.2 CRTC 1

La VSYNC débute lorsque C4=R7.

Si R7 est modifié avec la valeur de C4, alors la VSYNC est déclenchée immédiatement.
Si R7 est modifié sur C0vs=#36, par exemple, alors la prochaine lecture du PPI (au plus tôt 5 µsec après, donc sur C0=#3B) renvoie un statut actif de la VSYNC.

L'activation "déclenchée" de la VSYNC comptabilise la ligne comme si la VSYNC avait débuté en C0=0.

En conséquence, la durée totale de la VSYNC est réduite du nombre de µsec correspondant à la valeur de C0+1 du moment où R7 a été mis à jour.
Si une VSYNC est déclenchée en cours de ligne numéro 1, alors la VSYNC se termine à la fin de la ligne 16.

15.2.3 CRTC 2

La VSYNC est prise en compte sur toutes les valeurs de C0 et C9 lorsque C4=R7, mais également sur C0=R0 de la ligne précédent C4=R7 (lorsque C9=R9). Si une HSYNC a lieu lorsque la première condition de VSYNC est évaluée, alors le CRTC génère une VSYNC FANTOME.

Si R7 est modifié avec la valeur de C4, alors la VSYNC est déclenchée immédiatement, sauf durant la HSYNC, qui déclenche cette VSYNC FANTOME.

Une VSYNC FANTOME signifie que le CRTC comptabilise les lignes comme si une VSYNC avait lieu en empêchant une nouvelle VSYNC de se produire, mais sans que la broche VSYNC soit activée.

L'activation "déclenchée" de la VSYNC comptabilise la ligne comme si la VSYNC avait débuté en C0=0.

En conséquence, la durée totale de la VSYNC est réduite du nombre de µsec correspondant à la valeur de C0+1 du moment où R7 a été mis à jour.

Si une VSYNC est déclenchée en cours de ligne numéro 1, alors la VSYNC se termine à la fin de la ligne 16.

Pour contourner la problématique d’absence de VSYNC sur ce CRTC, il suffit juste d’éviter de générer une VSYNC FANTOME. Ainsi, en positionnant R7 loin dans le cosmos (par exemple 127), il suffit ensuite de mettre à jour R7 avec C4 lorsque C0 n’est plus présent dans la période de HSYNC de la ligne considérée.

15.2.4 CRTC 3, 4

La VSYNC débute lorsque C4=R7 et C9=C0=0.

Si R7 est modifié avec la valeur de C4 alors que C0>0, cela ne déclenchera pas de VSYNC.

15.3 LE BON MOMENT…

Si R7 est mis à jour sans « précautions », une nouvelle VSYNC peut survenir durant ou après le frame courant. Dans cette situation, le moniteur doit gérer plusieurs VSYNC ou l’absence de VSYNC après la mise à jour, et cela se traduit par des sauts d’images.

Le moniteur essaie simplement de caler son image sur la nouvelle position de C4=R7.
Cette opération peut avoir lieu plus ou moins vite selon le réglage « v-hold » du moniteur.

Il est possible d'éviter ce décrochage de synchronisation du moniteur en agissant pour que C4 repasse à la nouvelle valeur de R7 au même endroit que l'ancienne valeur de C4=R7. Cela nécessite de modifier R4 pour qu'il repasse à 0 plus tôt si le nouveau C4=R7 à atteindre est plus élevé que l'ancien, ou inversement d'agrandir R4 afin que C4 repasse à 0 plus tard si le nouveau C4=R7 à atteindre est moins élevé que l'ancien. Il faut juste imaginer que ce sont les compteurs qui viennent se placer où ils devraient se trouver.

Cette petite gymnastique de repositionnement des compteurs évite au moniteur de perdre la VSYNC, ce qui peut être ennuyeux pour un programme ne pouvant souffrir ce type d'artefact visuel.

A défaut, si votre code a le temps, il est toujours possible de dissimuler ce décrochage en agissant sur R1 et/ou R6, voir même en mettant toutes les encres noires un « certain temps », mais il existe quelques pervers qui titillent le V-Hold de leur moniteur afin de traquer les déviants insoumis du diktat des puristes intégristes.

16 SYNCHROS : REGISTRE R2

16.1 GÉNÉRALITÉS

Le registre CRTC R2 définit quand se produit la **HSYNC**.
La **HSYNC** se produit par rapport à la position du caractère **C0vs** (au sens CRTC).
Pendant la **HSYNC**, en principe, plus rien n'est affiché.

La longueur en µsec de la HSYNC-CRTC est fixée avec les 4 bits de poids faible du registre R3.
Voir chapitre 14.1, page 84.

Le moment ou se produit la HSYNC est différent selon les CRTC et n'est pas calé sur le début d'un caractère CRTC. Voir chapitre 14.4, page 85.

L'affichage de pixels ne débute pas (et ne s'arrête pas) sur une frontière de mot, voir d'octet (selon conditions) pour la HSYNC. Voir chapitre 14.8, page 89.

Le GATE ARRAY est plus rapide pour gérer la HSYNC que pour afficher les caractères lus par les CRTC 0, 1, 2. La HSYNC débute donc plus tôt et est "visible" sur le caractère précédent celui pointé par R2, qui n'a pas encore été affiché.

Les ASIC (CRTC 3 et 4) gèrent une HSYNC en cohérence avec la valeur de C0 affichée, en retardant l'affichage de la HSYNC de 1 µsec.

Exemples :

- Si R2 vaut 10, sur un CPC avec CRTC (0, 1, 2) alors la HSYNC débute à partir de C0 affiché=9 (R2-1) sur une longueur de R3 µsec.
- Si R2 vaut 10, sur un CPC avec "CRTC" (3, 4) alors la HSYNC débute à partir de C0 affiché=10 sur une longueur R3 µsec.

A cause de ce décalage, l'étalonnage d'un moniteur CM14 (464+/6128+) est différente de celui d'un CTM 640/644 (464/664/6128).

Cependant le CRTC 4 n'est pas livré avec un CM14, mais il se comporte comme un CPC+ au niveau du signal HSYNC. Branché sur un CTM 640/644, l'image est donc décalée à gauche car la HSYNC se produit 1 µsec plus tard. Afin d'assurer une compatibilité visuelle avec les autres CRTC, il est possible d'en tenir compte en programmant R2 avec une valeur inférieure de 1 à celle programmée pour les autres CRTC.

Inversement, brancher un CPC CRTC 0, 1, 2 sur un moniteur CM14 devrait provoquer un décalage à droite de l'image.

Sur un moniteur CTM, le premier caractère visible à gauche est le 15ème caractère à partir de la position C0=R2 si R3>5. (Si R2=49 (avec R0=63), le 1er caractère visible est C0=63).

Cette différence entre la prise en compte et l'affichage ne change rien au comportement de gestion du compteur du CRTC lorsque R2 et R3 sont modifiés.

Nous l'avons vu, le GATE ARRAY est plus lent pour afficher les caractères que pour prendre en considération le signal HSYNC, mais il existe également **un délai entre l'instruction du Z80A qui met à jour le CRTC et la vitesse de ce dernier pour en tenir compte**.

Ainsi, si R2 est modifié très exactement sur le caractère (CRTC) visé, le CRTC reçoit l'information légèrement plus tard, traduisant ici les délais internes propres à l'instruction OUT du Z80A. Cela permet de réduire légèrement la taille de la HSYNC « affichée » Voir chapitre 14.2, page 85.

Le CRTC dispose d'une période pendant laquelle il "accepte" de prendre en considération R2.

Le GATE ARRAY dans cette situation, reçoit le signal HSYNC légèrement plus tard, ce qui a un impact sur la longueur "**affichée**" de cette dernière.

Le traitement HSYNC débute avec le traitement HSYNC-GA, qui dure au maximum 6 µsec.
C'est sur la HSYNC-GA que le moniteur se synchronise pour chaque ligne.

Si cette HSYNC change de position ou de longueur (et si cette longueur dépasse 2 µsec) plusieurs fois en cours de balayage cela entraine une distorsion de l'image par le moniteur.

Pendant les 2 premières µsec le GATE ARRAY n'envoie pas de signal de synchronisation (H/Csync) au moniteur. La HSYNC-GA est dépendante de la HSYNC-CRTC (début et longueur):

- Elle débute en même temps que la HSYNC du CRTC.
- Elle s'arrête en même temps si la HSYNC du CRTC dure moins de 6 µsec.

Une HSYNC de 2 µsec ne provoque pas d'envoi de signal de synchronisation au moniteur.
Au-dessus de cette durée de 2 µsec, le moniteur va essayer de synchroniser la ligne affichée.

Des effets de distorsion peuvent apparaitre :

- Si la HSYNC est trop courte (>2 µsec et < 6 µsec).
- Si plusieurs HSYNC de longueur > 2 µsec sont générées sur une même ligne.
- Si la ou les HSYNC ne sont pas alignées verticalement.

Durant une HSYNC, les différents CRTC ne gèrent plus la condition C0=R2, ce qui interdit à une HSYNC de redémarrer au sein d'une HSYNC.

Le CRTC 2 est le champion dans la différence de gestion de certaines conditions survenant sur des valeurs précises de C0 pendant une HSYNC:

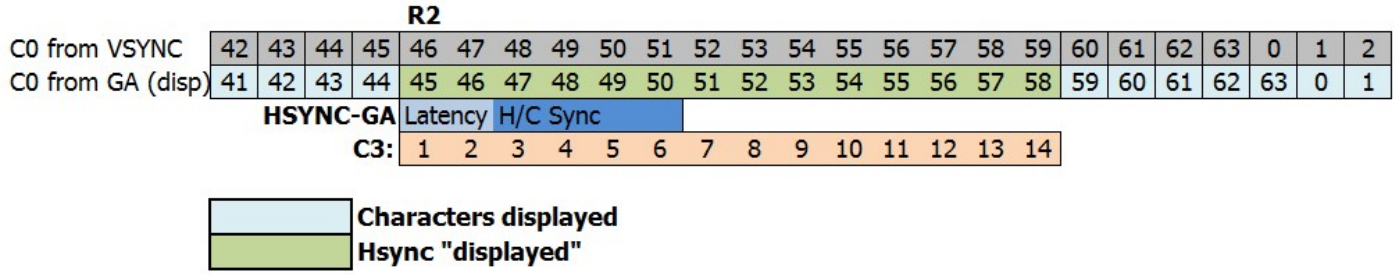
- C'est le cas lorsque C0=0 et que C4=R7 pour générer une VSYNC. Si une HSYNC a lieu durant cette période, alors la VSYNC est considérée comme activée.
- C'est le cas lorsque C0=0 pour que le BORDER soit désactivé. Une HSYNC placée sur la zone ou C0=0 ne permet pas de désactiver le BORDER avant un prochain C0=0.
- Une condition de dernière ligne effective durant la HSYNC force C4 à s'incrémenter au lieu de passer à 0.

16.2 HSYNC LORSQUE R2 EST PRÉDÉFINI

Les schémas suivants montrent la génération de la HSYNC, telle qu'elle survient lorsque R2 a été programmé avant que C0vs=R2, ce qui est en principe le cas général.
Les caractères sont indiqués selon les deux time-lines, puisque la HSYNC est gérée sans « délai » par le GATE ARRAY.

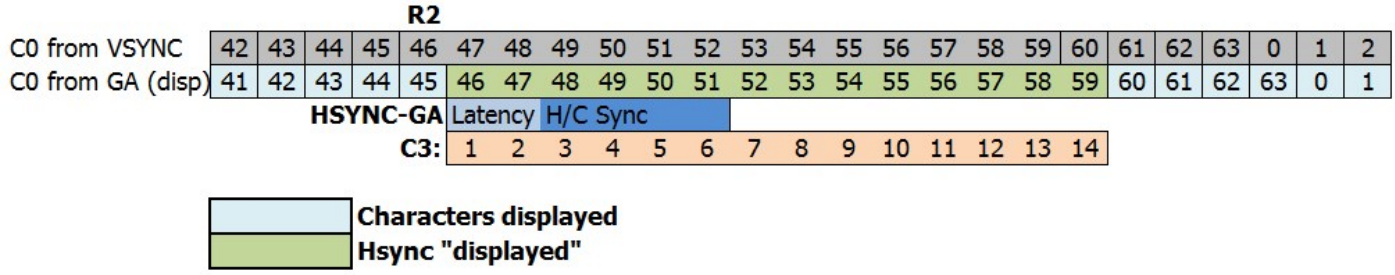
CRTC 0, 1, 2

CRTC-R2=46
CRTC-R3=14



CRTC 3, 4

CRTC-R2=46
CRTC-R3=14



16.3 MISE A JOUR DE R2 DURANT LA HSYNC

Durant le traitement de la HSYNC CRTC, une mise à jour de R2 n'est plus prise en compte si cette modification a pour objet de débuter une nouvelle HSYNC durant la HSYNC.

Remarque :

- Cette absence de prise en compte évite un plantage du circuit, qui ne génèrerait plus que des HSYNC dans la situation ou R0 serait inférieur à R3. C'est notamment le cas lorsque R0 est inférieur à R3, ce qui implique que C0 peut repasser plusieurs fois sur la même valeur (égale à R2).
- Sur le CRTC 0, deux HSYNC ne peuvent pas être collées, alors qu'il y a un bug de gestion sur les autres CRTC dans ce cas précis (ils évaluent mal C3=R3)

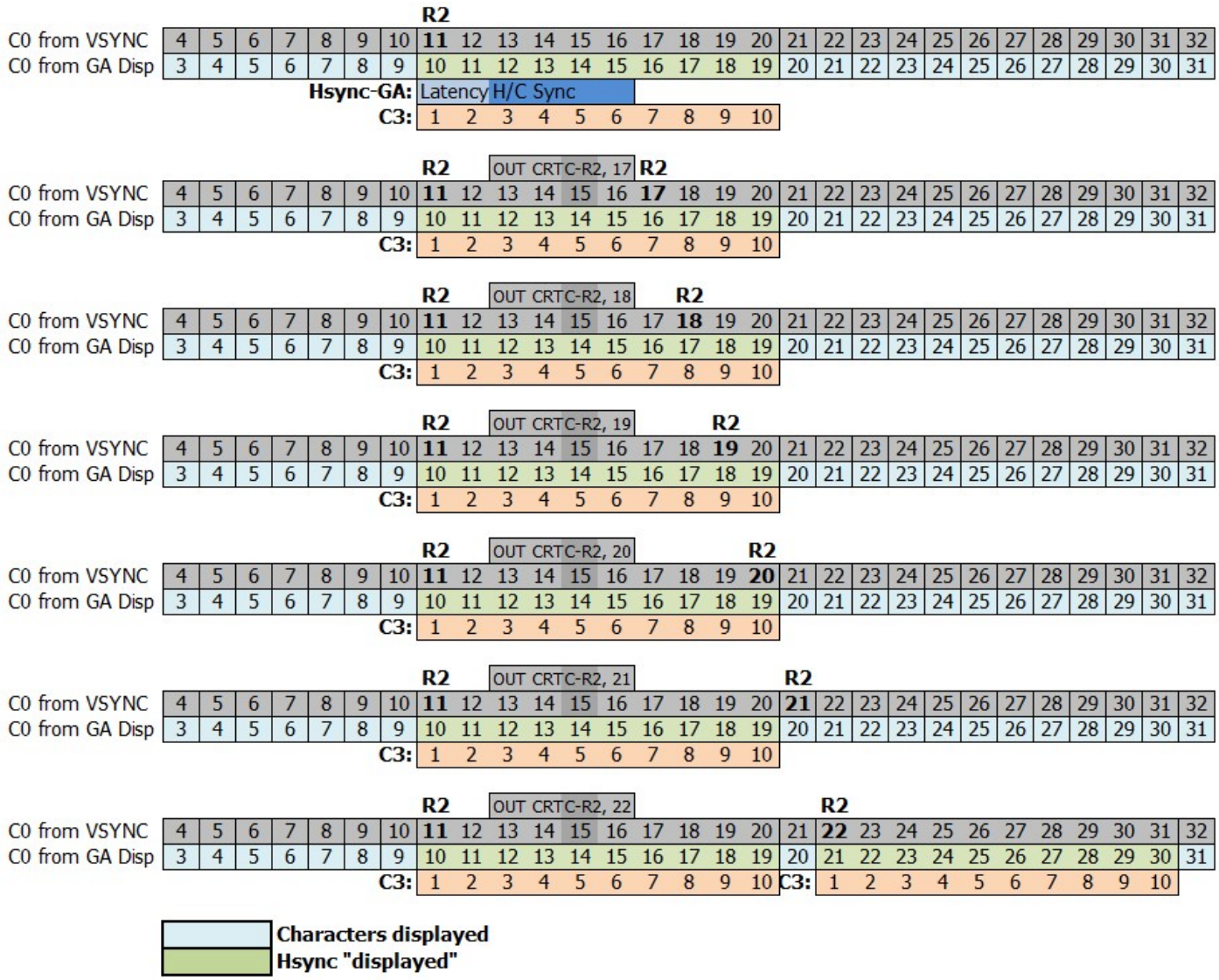
A tester en mode machiavel : sur CRTC 1, 2, 3, 4, placer R0=15, placer R3=0 (16 µsec) et R2=1.

Il y a quelques différences de prise en compte de R2 selon les CRTC, qui sont détaillées sur les schémas ci-après, qui décrivent la mise à jour de R2 sur différentes valeurs de **C0vs** durant la HSYNC, et l'impact sur cette dernière.

CRTC 0

CRTC-R2=11

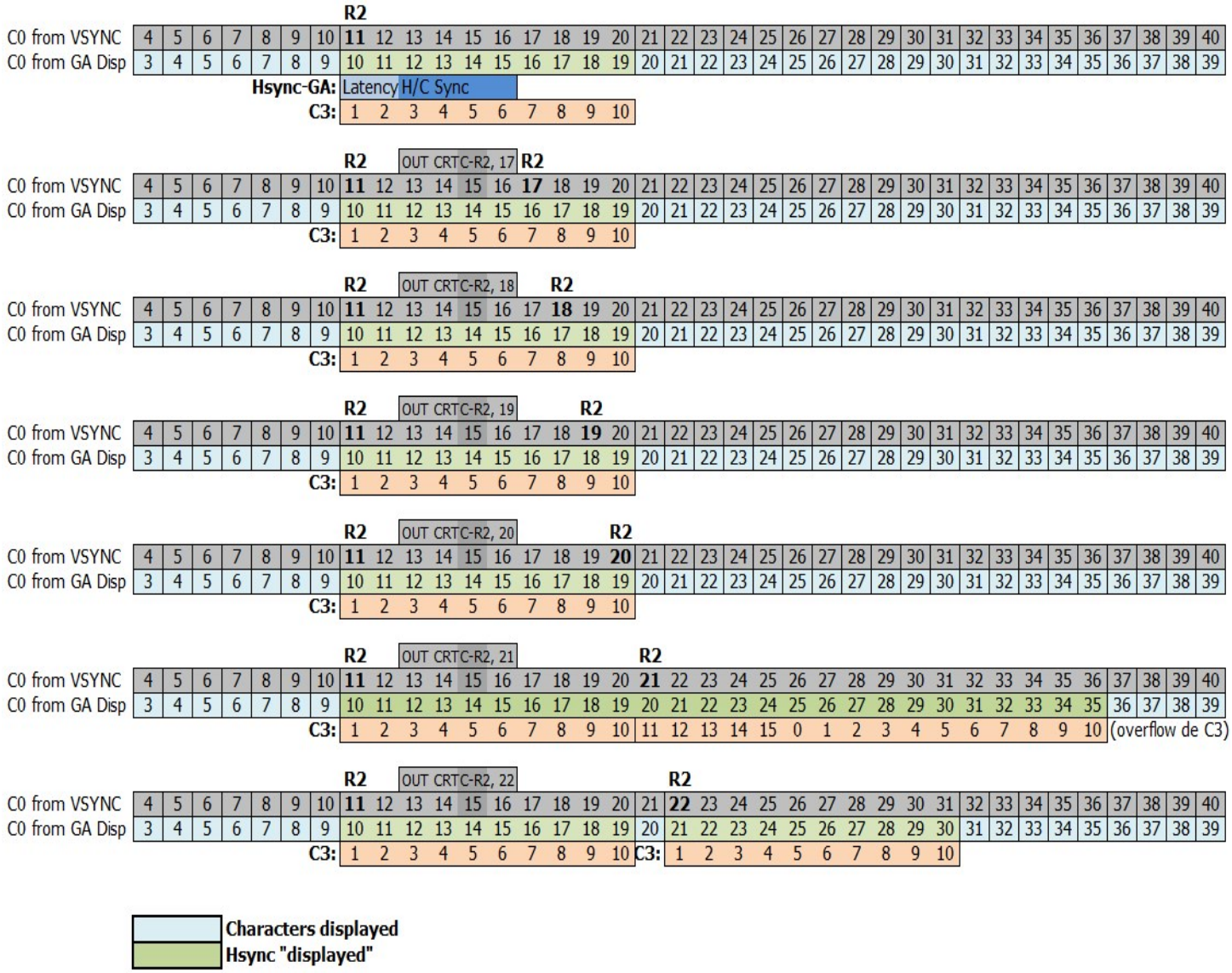
CRTC-R3=10 (HBL Size=10 chars)



CRTC 1, 2

CRTC-R2=11

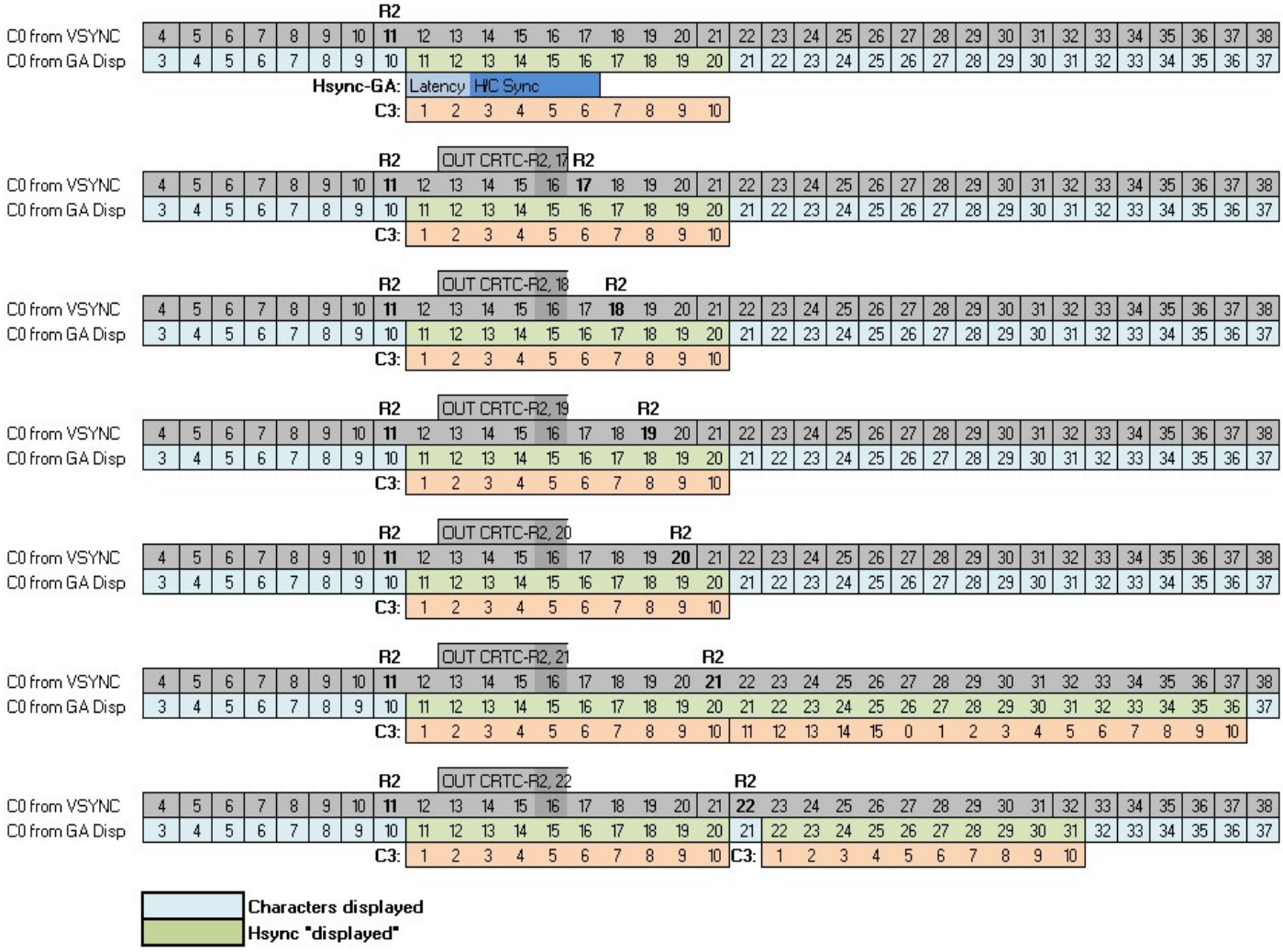
CRTC-R3=10 (HBL Size=10 chars)



CRTC 3, 4

CRTC-R2=11

CRTC-R3=10 (HBL Size=10 chars)



16.4 PRISE EN COMPTE VSYNC DURANT LA HSYNC

16.4.1 GÉNÉRALITÉS

L'évaluation de la condition VSYNC est réalisée quelque soit C0 pour les CRTC 0, 1 et 2, mais **uniquement lorsque C0=0 pour les CRTC 3 et 4**.

Dans tous les cas, C4 doit être égal à R7, soit parce que C4 y arrive, soit parce que R7 a été programmé avec la valeur de C4. C'est vrai également lorsque C4 atteint R7 à partir d'une gestion de ligne additionnelle.

Pour tous les CRTC sauf le 2, une condition de VSYNC se produisant durant la HSYNC ne pose aucun problème particulier. Voir le paragraphe ci-après pour les détails concernant le CRTC 2.

La valeur de R2 a été appliquée sur un frame complet dans les schémas suivants et décrit des HSYNC de différentes longueurs qui empiètent sur la zone de test de VSYNC.

La partie de la ligne sur les schémas est celle qui correspond à C4=R7-1 (ou précédent si R7 vaut 0), C9=7 (lorsque R9=7).

16.4.2 CRTC 0, 1

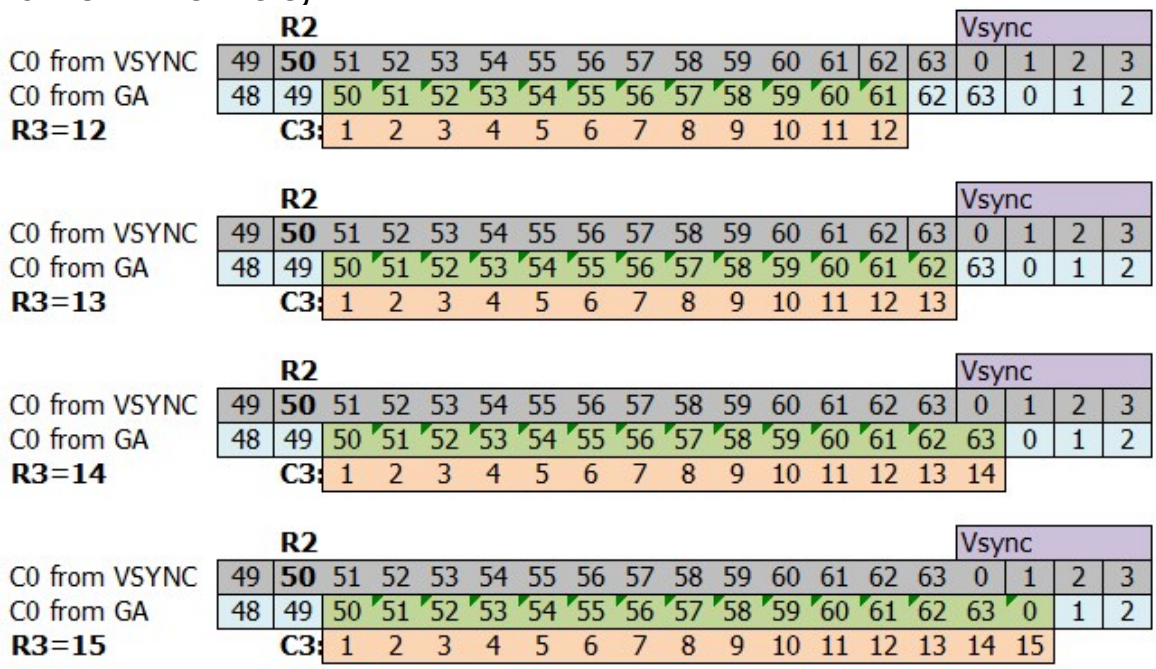
	R2														Vsync			
C0 from VSYNC	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2	3
C0 from GA	48	**49**	50	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2
R3=12	**C3:**	1	2	3	4	5	6	7	8	9	10	11	12						

	R2															Vsync			
C0 from VSYNC	49	**50**	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2	3
C0 from GA	48	**49**	50	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2
R3=13	**C3:**	1	2	3	4	5	6	7	8	9	10	11	12	13					

	R2															Vsync			
C0 from VSYNC	49	**50**	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2	3
C0 from GA	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2
R3=14	**C3:**	1	2	3	4	5	6	7	8	9	10	11	12	13	14				

	R2																Vsync			
C0 from VSYNC	49	**50**	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2	3
C0 from GA	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2
R3=15	**C3:**	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			

16.4.3 CRTC 3, 4



16.4.4 CRTC 2

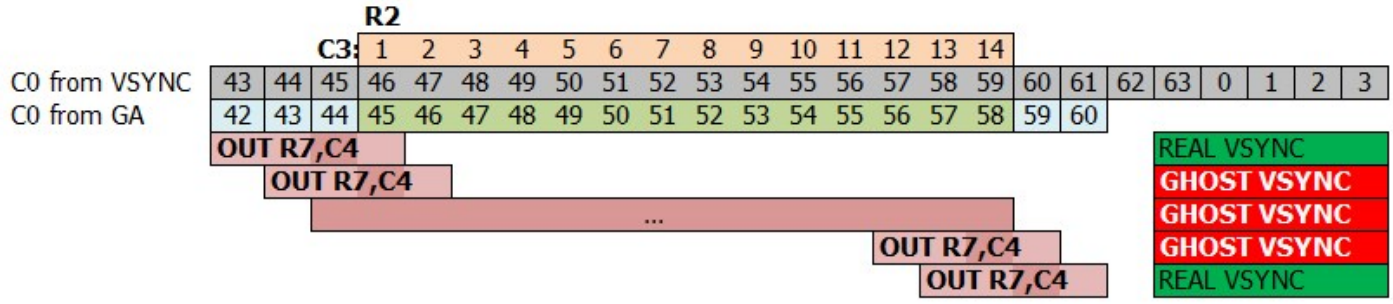
La condition VSYNC (C4=R7) est évaluée dès que C0=R0 sur la dernière ligne avant que C4 soit égal à R7. Elle est également évaluée pour toutes les valeurs de C0 pendant lesquelles C4=R7. Si cette première évaluation a lieu lorsque la HSYNC est en cours (lorsque la valeur de C0 ou se produit la condition est entre R2 et R2+R3-1), alors une VSYNC FANTÔME débute.

En effet, si la condition VSYNC était seulement « ignorée » durant la HSYNC, elle se produirait immédiatement en sortie de HSYNC, puisque la condition C4=R7 est encore vraie et l'évaluation a lieu sur toutes les valeurs de C9 et C0.

Il se produit très certainement un joli conflit sur l'activation de la broche VSYNC (broche 40) lorsque la broche HSYNC (broche 39) est à l'état haut.

La VSYNC ne peut plus survenir car le CRTC a activé son compteur de HSYNC et ne pourra donc accepter une nouvelle condition de VSYNC que lorsque la VSYNC FANTOME sera terminée.

Le schéma suivant décrit précisément l'inhibition de la VSYNC lorsque R7 est modifié avec le C4 courant.



Les schémas suivants décrivent les différentes valeurs de C0 sur lesquelles la condition VSYNC est évaluée en fonction de R2 et R3.

		R2														Vsync			
C0 from VSYNC	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2	3
C0 from GA	48	**49**	50	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2
R3=12	**C3:**	1	2	3	4	5	6	7	8	9	10	11	12						

		R2														Vsync			
C0 from VSYNC	49	**50**	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2	3
C0 from GA	48	**49**	50	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2
R3=13	**C3:**	1	2	3	4	5	6	7	8	9	10	11	12	13					

		R2														No Vsync			
C0 from VSYNC	49	**50**	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2	3
C0 from GA	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2
R3=14	**C3:**	1	2	3	4	5	6	7	8	9	10	11	12	13	14				

		R2														No Vsync			
C0 from VSYNC	49	**50**	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2	3
C0 from GA	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2
R3=15	**C3:**	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			

Il est possible de contourner la limitation du CRTC 2 à traiter sa VSYNC de plusieurs manière si on souhaite positionner R2 et R3 librement.

Une solution consiste à gérer soi-même R7, afin que la VSYNC ne puisse pas être traitée durant la HSYNC, et positionner R7 avec C4 une fois la HSYNC dépassée.

La FAKE VSYNC évoquée dans le chapitre 7.3 ne fonctionne pas correctement sur tous les CPC que j'ai pu tester. C'est donc une solution à éviter tant qu'on ne sait pas à quoi cette différence est liée et si on peut y pallier. De mes observations, ce n'est pas la mise à 1 du bit 0 du port B du PPI qui génère une VSYNC, mais plutôt le moment précis ou le CRTC annule la VSYNC fantôme.

Une autre solution consiste à modifier la taille de la HSYNC au bon moment, mais il existe une subtilité. Le résultat suivant est donc présenté sous réserve, car je n'ai pas encore compris pourquoi la VSYNC n'est pas générée lorsque R3 passe de 15 à 12, alors qu'elle l'est pour R3 qui passe de 14 à 12 :

La valeur de R3 est réduite au moment ou la condition VSYNC doit être évaluée.
Dans cette situation :

- Si R3 passe de 14 à 12 durant la HSYNC, la VSYNC se produit.
- Si R3 passe de 15 à 12 durant la HSYNC, la VSYNC ne se produit pas

		R2						OUT R3, 12								Vsync			
C0 from VSYNC	49	**50**	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2	3
C0 from GA	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2
R3 init=14	**C3:**	1	2	3	4	5	6	7	8	9	10	11	12	*13*	*14*				

		R2						OUT R3, 12								No Vsync			
C0 from VSYNC	49	**50**	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2	3
C0 from GA	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	0	1	2
R3 init=15	**C3:**	1	2	3	4	5	6	7	8	9	10	11	12	*13*	*14*	*15*			

16.5 DISPEN ET HSYNC

16.5.1 CRTC 0, 1, 3, 4

La gestion d'affichage du fond ou du BORDER, avec les conditions C0=0 et C0=R1, est prise en compte pour tous les CRTC sauf le 2.

16.5.2 CRTC 2

La condition permettant de rétablir l'affichage du fond a lieu lorsque C0=0 (et que C4 n'a jamais atteint R6).
Le BORDER est activé lorsque C0=R1.
Cependant, durant la HSYNC, ce test C0=0 n'est pas effectué.
(à voir si c'est la même chose avec C0=R1 pour activer le BORDER)

Dans cette condition, **le BORDER n'est pas désactivé**.
Ce qui est BORDERLINE, on peut le dire !

16.6 LE BON MOMENT…

Si R2 est mis à jour sans « précautions », une nouvelle HSYNC peut survenir durant ou après la fin de la ligne courante. Dans cette situation, le moniteur doit gérer plusieurs HSYNC ou l'absence de HSYNC après la mise à jour, et cela se traduit par une distorsion horizontale de l'image.

Le moniteur essaie de caler son image sur la nouvelle position de C0=R2.
Etant donné qu'il met plusieurs lignes pour y parvenir, cela se traduit, visuellement, par un décalage progressif des lignes. C'est à peu près le même principe que lorsque la longueur de la HSYNC est modifiée avec une valeur inférieure à 6 pour décaler l'image.

De nombreuses démos ont utilisé ces principes (R2 et/ou R3) pour effectuer des déformations horizontales de l'image ou réaliser des scrollings.

Tout comme pour R7 au niveau vertical, il est possible d'éviter ce décrochage de synchronisation horizontale du moniteur en agissant pour que C0 repasse à la nouvelle valeur de R2 au même endroit que l'ancienne valeur de C0=R2. Cela nécessite de modifier R0 pour qu'il repasse à 0 plus tôt si le nouveau C0=R2 à atteindre est plus élevé que l'ancien, ou inversement d'agrandir R0 afin que C0 repasse à 0 plus tard si le nouveau C0=R2 à atteindre est moins élevé que l'ancien. Il faut juste imaginer que ce sont les compteurs qui viennent se placer où ils devraient se trouver.

Cette petite gymnastique de repositionnement des compteurs évite au moniteur de perdre la HSYNC (ou d'en avoir deux en moins de 64 µsec), ce qui peut être ennuyeux pour un programme ne pouvant souffrir ce type d'artefact visuel.

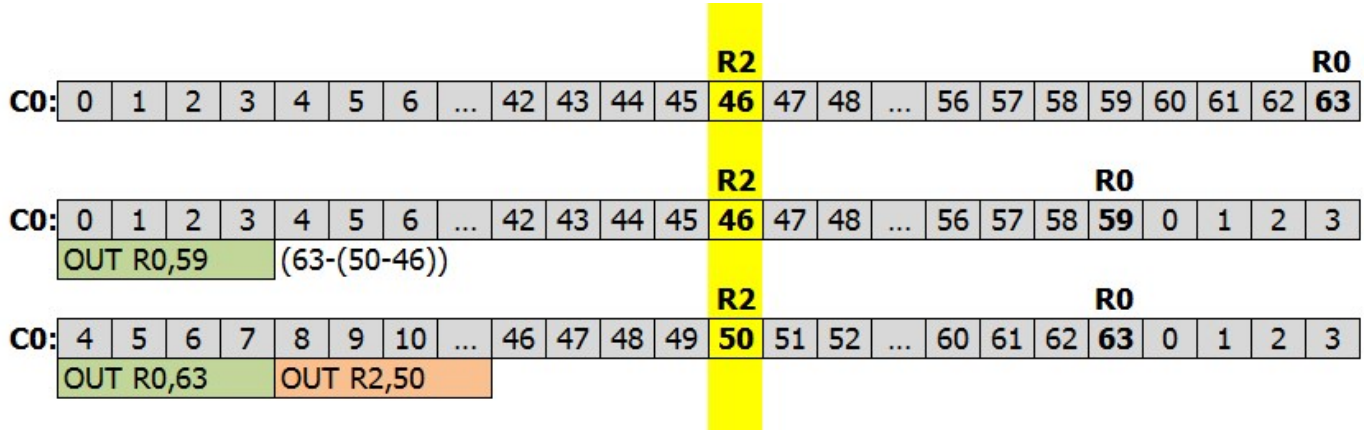
A défaut, si le CPC n'a pas vocation à servir le diktat de diffusion vidéo des organisateurs de gros meetings à concours (auquel est soumis le diktat évoqué dans le chapitre 15.3) ou s'adapter à de jolis écrans plats LCD, il est toujours possible de dissimuler ce décrochage en modifiant R2 dans une zone non affichée (par exemple durant la VSYNC).

Il est aussi possible, comme pour la synchro verticale, d'agir sur R1 et/ou R6, ou même mettre toutes les encres noires un « certain temps ».

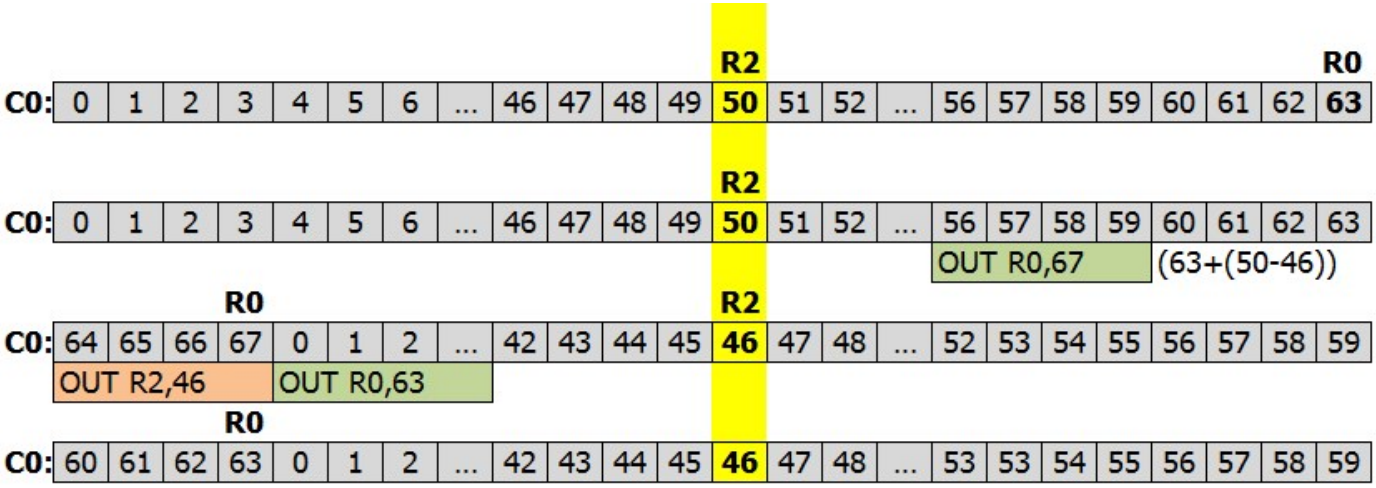
Le réglage du potentiomètre H-Hold nécessite toutefois un petit tournevis plat pour être réglé sur un moniteur CTM.

Voici néanmoins 2 schémas pour traduire la gymnastique des compteurs nécessaire pour passer de R2 de 46 à 50 et inversement, de 50 à 46 (le choix de ces valeurs n'étant ni fortuit, ni indépendant de ma volonté) sans traumatiser l'écran.

16.6.1 Passer de R2=46 à R2=50 sur des lignes de 64 µsec



16.6.2 Passer de R2=50 à R2=46 sur des lignes de 64 µsec



17 AFFICHAGE : REGISTRE R1

17.1 GÉNÉRALITÉS

Ce registre a pour fonction de définir le nombre de caractères horizontaux affichés sur une ligne.
Sa valeur est exprimée en nombre de caractères CRTC (pour rappel, chaque caractère CRTC occupe 2 octets en RAM).

Il joue également un rôle important pour la mémorisation du pointeur vidéo courant.

De manière générale, le CRTC génère R0+1 caractères, dont R1 caractères seront affichés.

Lorsque R1 caractères ont été affichés (DISPLAY ENABLE ON) alors l'affichage est inhibé, et du "BORDER" est affiché sur CPC (DISPLAY ENABLE OFF) via le GATE ARRAY.

A noter que la broche DISPLAY ENABLE porte divers nom, et on pourra la retrouver sous d'autres appellations exotiques, comme DISPTMG ou DE selon les différentes documentations CRTC.

Pour afficher l'intégralité des caractères que le CRTC peut générer pour une ligne, R1 devrait en principe valoir R0+1. En effet, le "flag" DISPLAY ENABLE est positionné à OFF pour générer du BORDER lorsque C0=R1.

Or, C0 ne peut jamais dépasser R0.
Pour afficher tous les caractères programmés avec R0, il est possible d'empêcher que C0 atteigne R1, soit en positionnant R1>R0, soit en changeant la valeur de R1 en cours de ligne.

Dans cette situation, un problème se pose car l'égalité entre C0 et R1 sert à la mise à jour du pointeur vidéo lorsque C9=R9 (dernière ligne d'un "caractère").

Ce défaut de mise à jour provoque dès lors une répétition des lignes de caractères, dans des conditions propres à chacun des CRTC.

Remarque :
Ce défaut lié à la gestion dynamique des compteurs (dite de bornes et de poteaux) est une plaie car il ne permet pas au compteur vidéo d'atteindre 128 octets pour 1 ligne (64 µsec), ce qui peut conduire certains malheureux à utiliser des écrans de 65 µsec par ligne en positionnant R0 à 64 avec une position de synchro horizontale unique pour ces lignes de 65µsec...
Cela permet d'ajouter un sifflement aux musiques jouées dans les démos et vérifier que le moniteur est mal réglé.
Sans artifice cela ne permet pas de disposer d'un pointeur vidéo dont le poids fort ne varie pas durant l'affichage d'une ligne. Ce « truc » permet de gagner de la CPU lorsqu'il est question d'afficher des données, car on peut se contenter d'incrémenter un registre 8 bits au lieu de 16 bits pour le pointeur vidéo. C'est pourquoi, malheureusement, tant d'écrans sont formatés avec des lignes de 64 octets de large en « minimized screen », soit 16 octets de moins que la largeur standard.

Lorsque C0 repasse à 0 (condition C0=R0) alors l'affichage est autorisé (DISPLAY ENABLE ON).
A noter cependant que **cette condition n'est pas gérée par le CRTC 2 durant la HSYNC.**

Si R1 est positionné à 0, alors plus aucun caractère n'est affiché, quel que soit le CRTC d'un CPC.

Remarque :
A priori ce ne serait pas le cas sur le BBC avec le Hitachi HD6845SP (type 0) soit avec Samsung KS68C45S ou un VLSI VL68C45S23PC. (mais pourquoi je parle de ça dans un document dédié au CPC, moi ?). (pourquoi pas le NEC PD7220 tant que j'y suis ?)

Dans les schémas ci-après, et dans quelques propos liminaires, je fais référence à deux pointeurs mémoire dans le CRTC, que j'ai nommé **VMA** et **VMA'**.

Lorsque le CRTC affiche des caractères, il se sert toujours du pointeur **VMA**.

Ce pointeur est incrémenté à chaque fois qu'un caractère est traité par le CRTC, qu'il soit affichable ou non. Dans certaines « préversions » de CRTC, ce pointeur n'était pas géré durant la VSYNC, mais cela ne concerne pas les CRTC des CPC à ma connaissance.

Lorsque C0=R1 et C9=R9, alors le pointeur courant **VMA** est transféré dans le pointeur **VMA'**.

Lorsque C0=0, à chaque début de ligne, le pointeur de ligne **VMA'** est transféré dans le pointeur courant **VMA**, sauf pour le premier caractère (C4=0).

17.2 AFFICHAGES SELON R1

17.2.1 AFFICHAGE AVEC R1 <= R0

Le schéma suivant décrit la gestion R1 dans un cadre de programmation "standard" du CRTC sur un CPC qui vient d'être allumé avec une rom Basic standard.

Données initiales :
CRTC-R0=63
CRTC-R1=40
CRTC-R9=7
CRTC-R12=0
CRTC-R13=0

													R1							R0
C0=0	**C0=R1 & C9=R9**	**VRAM-C9-Bit 0..2**		**C0:**	0	1	2	3	...	37	38	39	**40**	41	42	43	44	45	...	63
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	0		0	1	2	3	...	37	38	39	DISP-OFF							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	1		0	1	2	3	...	37	38	39	DISP-OFF							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	2		0	1	2	3	...	37	38	39	DISP-OFF							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	3		0	1	2	3	...	37	38	39	DISP-OFF			**BORDER**				
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	4		0	1	2	3	...	37	38	39	DISP-OFF							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	5		0	1	2	3	...	37	38	39	DISP-OFF							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	6		0	1	2	3	...	37	38	39	DISP-OFF							
CRTC-VMA=CRTC-VMA'	**CRTC-VMA'=CRTC-VMA**	CRTC-VMA C9 :	7		0	1	2	3	...	37	38	39	DISP-OFF							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	0		40	41	42	43	...	77	78	79	DISP-OFF							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	1		40	41	42	43	...	77	78	79	DISP-OFF							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	2		40	41	42	43	...	77	78	79	DISP-OFF							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	3		40	41	42	43	...	77	78	79	DISP-OFF							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	4		40	41	42	43	...	77	78	79	DISP-OFF							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	5		40	41	42	43	...	77	78	79	DISP-OFF							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	6		40	41	42	43	...	77	78	79	DISP-OFF							
CRTC-VMA=CRTC-VMA'	**CRTC-VMA'=CRTC-VMA**	CRTC-VMA C9 :	7		40	41	42	43	...	77	78	79	DISP-OFF							

17.2.2 AFFFICHAGE AVEC R1 > R0

Le schéma suivant décrit l'affichage lorsque R1 est programmé avec une valeur supérieure à 63.

Données initiales :
CRTC-R0=63
CRTC-R1=**64**
CRTC-R9=7
CRTC-R12=0
CRTC-R13=0

Si R1 > R0, alors C0 n'est jamais égal à R1 (lorsque C9=R9), et **VMA'** n'est donc pas mis à jour avec **VMA.**

Cela provoque une répétition de caractères car l'adresse courante n'est pas mise à jour lors du changement de ligne-caractère (lorsque C9=R9).

Remarque 1 :
Seuls les 10 premiers bits ne sont pas mis à jour dans ce contexte.
Les bits qui déterminent le numéro de bloc (ligne caractère) (C9 ou C5) continuent de "participer" à l'adresse. Voir chapitre 20, page 149.

Remarque 2 :
Dans la mesure où les conditions permettent une prise en compte de l'adresse, modifier R12/R13 permet d'éviter cette répétition. En pratique cela permet donc de coller les "lignes" lorsque C0 repasse à 0 plusieurs fois durant une "ligne". Cependant les CRTC 0 et 2 génèrent un octet de BORDER, sachant que le CRTC 0 peut tout de même empêcher la génération de cet octet en consommant de la CPU. Voir chapitre 19.2, page 124.

Remarque 3 :
Selon les CRTC, la mise à jour initiale de CRTC-VMA'/CRTC-VMA via R12/R13 n'est pas la même.
Voir chapitres suivants et chapitre 19.2, page 124.

C0=0	C0=R1 & C9=R9	VRAM-C9-Bit 0..2	
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9:	0
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9:	1
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9:	2
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9:	3
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9:	4
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9:	5
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9:	6
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9:	7
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9:	0
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9:	1
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9:	2
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9:	3
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9:	4
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9:	5
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9:	6
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9:	7

R0 R1

C0:	0	1	2	3	...	53	54	55	56	57	58	59	60	61	62	63
0	0	1	2	3	...	53	54	55	56	57	58	59	60	61	62	63
1	0	1	2	3	...	53	54	55	56	57	58	59	60	61	62	63
2	0	1	2	3	...	53	54	55	56	57	58	59	60	61	62	63
3	0	1	2	3	...	53	54	55	56	57	58	59	60	61	62	63
4	0	1	2	3	...	53	54	55	56	57	58	59	60	61	62	63
5	0	1	2	3	...	53	54	55	56	57	58	59	60	61	62	63
6	0	1	2	3	...	53	54	55	56	57	58	59	60	61	62	63
7	0	1	2	3	...	53	54	55	56	57	58	59	60	61	62	63
0	0	1	2	3	...	53	54	55	56	57	58	59	60	61	62	63
1	0	1	2	3	...	53	54	55	56	57	58	59	60	61	62	63
2	0	1	2	3	...	53	54	55	56	57	58	59	60	61	62	63
3	0	1	2	3	...	53	54	55	56	57	58	59	60	61	62	63
4	0	1	2	3	...	53	54	55	56	57	58	59	60	61	62	63
5	0	1	2	3	...	53	54	55	56	57	58	59	60	61	62	63
6	0	1	2	3	...	53	54	55	56	57	58	59	60	61	62	63
7	0	1	2	3	...	53	54	55	56	57	58	59	60	61	62	63

17.3 MISE JOUR DYNAMIQUE DE R1

La condition C0=R1 est prise en compte immédiatement sur une ligne.
Elle peut survenir plusieurs fois sur une même ligne si R1 est reprogrammé.

Lorsque la première condition C0=R1 est vraie, il n'y a plus d'affichage des caractères et du BORDER est affiché.

Cependant, même si du BORDER est affiché, le pointeur **VMA continue de compter** (Le CRTC n'étant qu'un vaste champ de compteurs en fleurs...).

Si R1 est modifié une nouvelle fois durant la ligne pour satisfaire de nouveau la condition C0=R1 lorsque C9=R9 alors cela **entrainera une mise à jour du pointeur vidéo**.

Dis autrement, la modification de R1 durant l'affichage de BORDER R1 permet de mettre à jour le pointeur vidéo **sans que les données soient affichées**.

Les schémas ci-après montrent ces comportements lorsque C4>0.

C0=0	C0=R1 & C9=R9	VRAM-C9-Bit 0..2		Upd C0:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	0	R1=26	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	DISP-OFF					
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	1	R1=25	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	DISP-OFF						
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	2	R1=24	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	DISP-OFF							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	3	R1=23	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	DISP-OFF								
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	4	R1=22	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	DISP-OFF									
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	5	R1=21	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	DISP-OFF										
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	6	R1=20	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	DISP-OFF											
CRTC-VMA=CRTC-VMA'	**CRTC-VMA'=CRTC-VMA**	CRTC-VMA C9 :	7	R1=19	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	DISP-OFF												
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	0	R1=18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	DISP-OFF													
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	1	R1=17	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	DISP-OFF														
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	2	R1=16	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	DISP-OFF															
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	3	R1=15	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	DISP-OFF																
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	4	R1=14	19	20	21	22	23	24	25	26	27	28	29	30	31	32	DISP-OFF																	
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	5	R1=13	19	20	21	22	23	24	25	26	27	28	29	30	31	DISP-OFF																		
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	6	R1=12	19	20	21	22	23	24	25	26	27	28	29	30	DISP-OFF																			
CRTC-VMA=CRTC-VMA'	**CRTC-VMA'=CRTC-VMA**	CRTC-VMA C9 :	7	R1=11	19	20	21	22	23	24	25	26	27	28	29	DISP-OFF																				
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	0	R1=12	30	31	32	33	34	35	36	37	38	39	40	41	DISP-OFF																			

R0

C0=0	C0=R1 & C9=R9	VRAM-C9-Bit 0..2		Upd C0:	0	1	2	3	...	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	**63**
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	0	R1=64	0	1	2	3	...	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	1		0	1	2	3	...	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	2		0	1	2	3	...	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	3		0	1	2	3	...	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	4		0	1	2	3	...	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	5		0	1	2	3	...	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	6		0	1	2	3	...	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
CRTC-VMA=CRTC-VMA'	**CRTC-VMA'=CRTC-VMA**	CRTC-VMA C9 :	7	R1=40	0	1	2	3	...	37	38	39	DISP-OFF																									
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	0		40	41	42	43	...	77	78	79	DISP-OFF																									
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	1		40	41	42	43	...	77	78	79	DISP-OFF																									
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	2		40	41	42	43	...	77	78	79	DISP-OFF																									
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	3		40	41	42	43	...	77	78	79	DISP-OFF																									
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	4		40	41	42	43	...	77	78	79	DISP-OFF																									
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	5		40	41	42	43	...	77	78	79	DISP-OFF																									
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	6		40	41	42	43	...	77	78	79	DISP-OFF																									
CRTC-VMA=CRTC-VMA'	**CRTC-VMA'=CRTC-VMA**	CRTC-VMA C9 :	7		40	41	42	43	...	77	78	79	DISP-OFF																									

													R1																							R0
C0=0	**C0=R1 & C9=R9**	**VRAM-C9-Bit 0..2**		**UpdC0:**	0	1	2	3	…	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	**63**
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	0	R1=40	0	1	2	3	…	37	38	39	DISP-OFF																							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	1		0	1	2	3	…	37	38	39	DISP-OFF																							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	2		0	1	2	3	…	37	38	39	DISP-OFF																							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	3		0	1	2	3	…	37	38	39	DISP-OFF																							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	4		0	1	2	3	…	37	38	39	DISP-OFF																							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	5		0	1	2	3	…	37	38	39	DISP-OFF																							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	6		0	1	2	3	…	37	38	39	DISP-OFF																							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	7	R1=64	0	1	2	3	…	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	0		0	1	2	3	…	37	38	39	DISP-OFF																							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	1		0	1	2	3	…	37	38	39	DISP-OFF																							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	2		0	1	2	3	…	37	38	39	DISP-OFF																							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	3		0	1	2	3	…	37	38	39	DISP-OFF																							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	4		0	1	2	3	…	37	38	39	DISP-OFF																							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	5		0	1	2	3	…	37	38	39	DISP-OFF																							
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	6		0	1	2	3	…	37	38	39	DISP-OFF																							
CRTC-VMA=CRTC-VMA'	**CRTC-VMA'=CRTC-VMA**	CRTC-VMA C9 :	7		0	1	2	3	…	37	38	39	DISP-OFF																							

																	R1																			R0
C0=0	**C0=R1 & C9=R9**	**VRAM-C9-Bit 0..2**		**UpdC0:**	0	1	2	…	32	33	34	35	36	37	38	39	40	41	42	43	44	45	…	51	52	53	54	55	56	57	58	59	60	61	62	**63**
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	0	R1=40	0	1	2	…	32	33	34	35	36	37	38	39	DISP-OFF																			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	1		0	1	2	…	32	33	34	35	36	37	38	39	DISP-OFF																			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	2		0	1	2	…	32	33	34	35	36	37	38	39	DISP-OFF																			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	3		0	1	2	…	32	33	34	35	36	37	38	39	DISP-OFF																			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	4		0	1	2	…	32	33	34	35	36	37	38	39	DISP-OFF																			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	5		0	1	2	…	32	33	34	35	36	37	38	39	DISP-OFF																			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	6		0	1	2	…	32	33	34	35	36	37	38	39	DISP-OFF																			
CRTC-VMA=CRTC-VMA'	**CRTC-VMA'=CRTC-VMA**	CRTC-VMA C9 :	7		0	1	2	…	32	33	34	35	36	37	38	39	DISP-OFF																**OUT R1,0**			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	0		DISP-OFF			**CRTC-VMA ++++**					**OUT R1,40**					**CRTC-VMA +++++**															**OUT R1,0**			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	1		DISP-OFF			**CRTC-VMA ++++**					**OUT R1,40**					**CRTC-VMA +++++**															**OUT R1,0**			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	2		DISP-OFF			**CRTC-VMA ++++**					**OUT R1,40**					**CRTC-VMA +++++**															**OUT R1,0**			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	3		DISP-OFF			**CRTC-VMA ++++**					**OUT R1,40**					**CRTC-VMA +++++**															**OUT R1,0**			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	4		DISP-OFF			**CRTC-VMA ++++**					**OUT R1,40**					**CRTC-VMA +++++**															**OUT R1,0**			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	5		DISP-OFF			**CRTC-VMA ++++**					**OUT R1,40**					**CRTC-VMA +++++**															**OUT R1,0**			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	6		DISP-OFF			**CRTC-VMA ++++**					**OUT R1,40**					**CRTC-VMA +++++**															**OUT R1,0**			
CRTC-VMA=CRTC-VMA'	**CRTC-VMA'=CRTC-VMA**	CRTC-VMA C9 :	7		DISP-OFF			**CRTC-VMA ++++**					**OUT R1,40**					**CRTC-VMA +++++**																			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	0		80	81	82	…	112	113	114	115	116	117	118	119	DISP-OFF																			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	1		80	81	82	…	112	113	114	115	116	117	118	119	DISP-OFF																			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	2		80	81	82	…	112	113	114	115	116	117	118	119	DISP-OFF																			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	3		80	81	82	…	112	113	114	115	116	117	118	119	DISP-OFF																			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	4		80	81	82	…	112	113	114	115	116	117	118	119	DISP-OFF																			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	5		80	81	82	…	112	113	114	115	116	117	118	119	DISP-OFF																			
CRTC-VMA=CRTC-VMA'		CRTC-VMA C9 :	6		80	81	82	…	112	113	114	115	116	117	118	119	DISP-OFF																			
CRTC-VMA=CRTC-VMA'	**CRTC-VMA'=CRTC-VMA**	CRTC-VMA C9 :	7		80	81	82	…	112	113	114	115	116	117	118	119	DISP-OFF																			

17.4 VMA'/VMA LORSQUE C4=0

Il existe des différences entre les CRTC pour l'affectation du pointeur vidéo avec les valeurs programmées dans R12 et/ou R13 lorsque C4=0 et C0=0.

17.4.1 CRTC 0, 3, 4

La première ligne caractère commence avec l'adresse définie par R12/R13, quelque soit la valeur de R1.
Sur la première ligne du premier caractère d'un "écran" (C4=C9=C0=0), **le pointeur VMA' est mis à jour à l'aide du contenu de R12/R13.**
Cette mise à jour est suivie de la **mise à jour du pointeur VMA avec VMA'**.

Si R1>R0, le pointeur VMA' n'est plus mis à jour et les lignes se répètent.

Dans cette circonstance, lorsqu'un nouvel écran débute, **VMA' est mis à jour et toutes les lignes affichées deviennent identiques et égales au pointeur R12/R13**.

17.4.2 CRTC 1

La première ligne caractère commence avec l'adresse définie par R12/R13, quelque soit la valeur de R1.
Lorsqu'on est sur le premier caractère d'un "écran" (C4=C0=0), **le pointeur VMA** (et non VMA' comme sur CRTC 0) **est mis à jour à l'aide du contenu de R12/R13**.

Remarque : cette particularité de mise à jour permet de modifier l'offset via R12 et/ou R13 sur chaque ligne (C9=0 à R9) d'un caractère pour lequel C4=0.

Cette mise à jour de VMA a cependant une conséquence lorsque R1>R0 durant tout le frame.

En effet, la condition C0=R1+1 ne se produit plus et le pointeur **VMA' n'est plus mis à jour**.
VMA' est "figé" sur le dernier pointeur connu lorsque C0 avait atteint R1+1 et lorsque C9=R9.
Lorsque C9=0, le pointeur VMA est rechargé avec VMA' lorsque C4>0.

On a donc, lorsque R1>R0, une première ligne caractère qui contient le pointeur défini dans R12/R13 et les suivantes le dernier pointeur mis à jour dans VMA'.

Le pointeur vidéo continue cependant d'être incrémenté, même lorsque du BORDER est affiché.

C'est vrai au niveau horizontal (gestion R1) mais aussi en vertical (gestion R6).

Lorsque le pointeur sur 10 bits continue de s'incrémenter, il atteint sa limite (même si ce compteur peut utiliser les Overscan Bits ™).

Exemple :
Si la largeur d'affichage d'une ligne est de 40 caractères (&28), le CRTC va pouvoir afficher 1024/40 lignes caractères différentes, soient 25 lignes complètes.

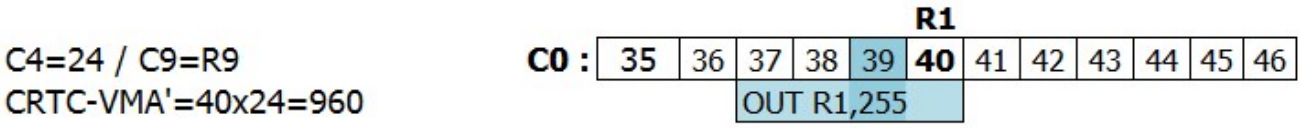
Lorsque C4=R6=25, le CRTC cesse d'afficher les lignes-caractères mais il continue cependant d'incrémenter son pointeur et de gérer les mises à jour de VMA' lorsque C0=R1 & C9=R9.

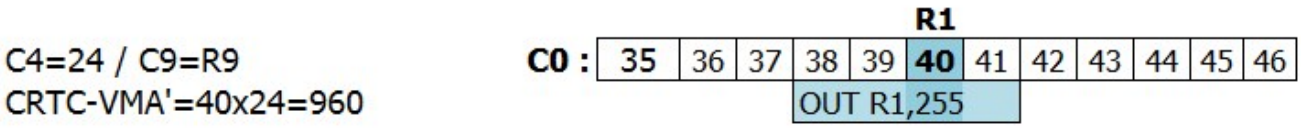
Si la modification de R1 (>R0) intervient durant cette période, le pointeur vidéo aura débordé.

Si R1 est modifié après que C4=24, C9=R9 et C0>R1, alors le pointeur sera égal à 40 x 25=1000 (la ligne répétée contiendra les caractères 1000 à 1023, suivi de 0000 à 0015).

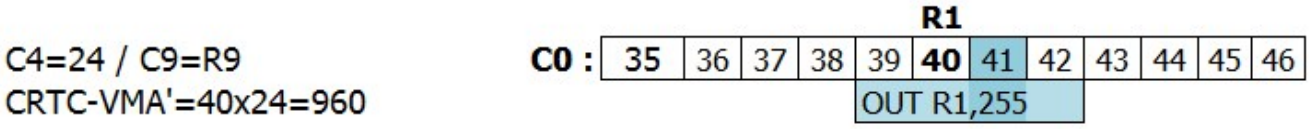
Si R1 est modifié après que C4=25, C9=R9 et C0>R1, alors le pointeur sera égal à 40 x 26=(1040 and 1023)=16 (la ligne répétée contiendra les caractères 0016 à 0055).

Pointeur VMA' répété = 960 :





Pointeur VMA' répété = 1000 (40 x 25) :
La condition C0=R1 a eu lieu, le pointeur VMA' a été chargé avec VMA



17.4.3 CRTC 2

Lorsque R1>R0, **aucun des deux pointeurs n'est mis à jour avec R12/R13.**
Toutes les lignes sont identiques dans cette situation.

Sur la première ligne, VMA est affecté par VMA' (qui a lui-même été affecté par R12/R13 lorsque C0=R1 sur la dernière ligne écran).

L'adresse répliquée suit la même logique que celle décrite pour le CRTC 1.

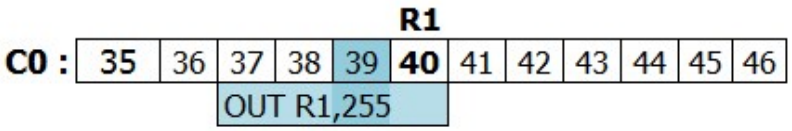
Donc, selon le moment ou R1 devient supérieur à R0, c'est le pointeur au moment ou le BORDER R1 est géré qui est pris en compte.

Il existe toutefois une différence avec le CRTC 1 sur le moment ou la modification de R1 est prise en compte.

Le CRTC 1 réalise l'affectation de VMA' avec VMA **lorsque C0=R1+1**.
Le CRTC 2 réalise cette même opération **lorsque C0=R1**.

Pointeur VMA' répété = 960 :

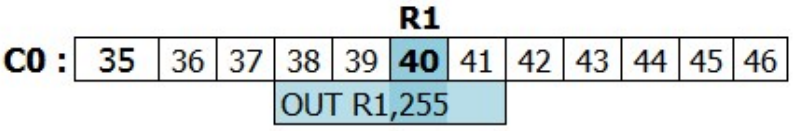
C4=24 / C9=R9
CRTC-VMA'=40x24=960



Pointeur VMA' répété = 1000 (40 x 25) :

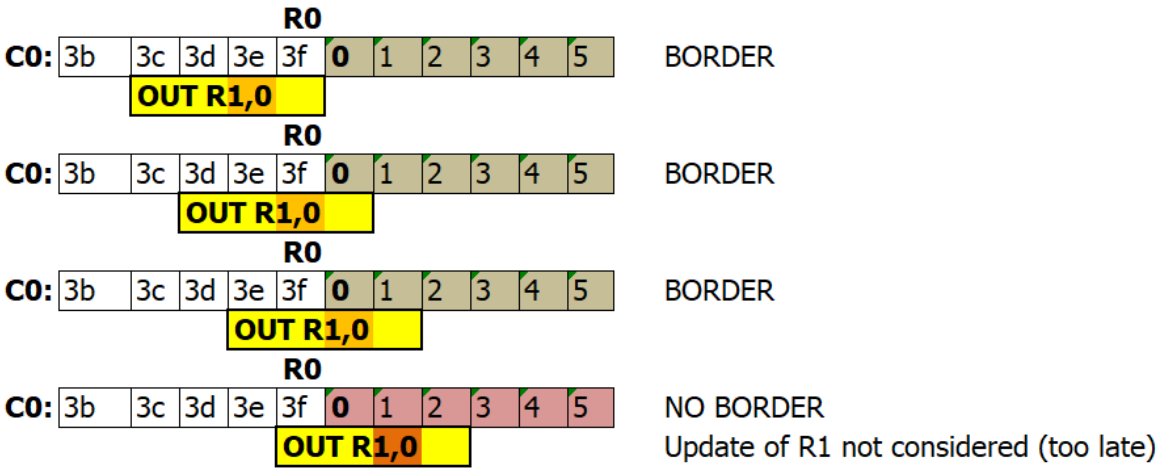
La condition C0=R1 a eu lieu, le pointeur CRTC-VMA' a été chargé avec CRTC-VMA

C4=24 / C9=R9
CRTC-VMA'=40x24=960



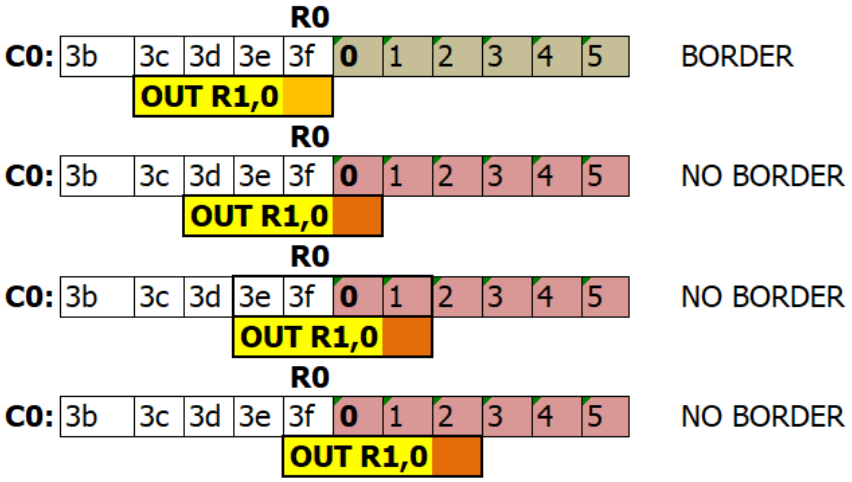
17.5 PRISE EN COMPTE R1=0

17.5.1 CRTC 0, 1, 2



Update of R1 not considered (too late)
Update of R1 ok (just in time)

17.5.2 CRTC 3, 4



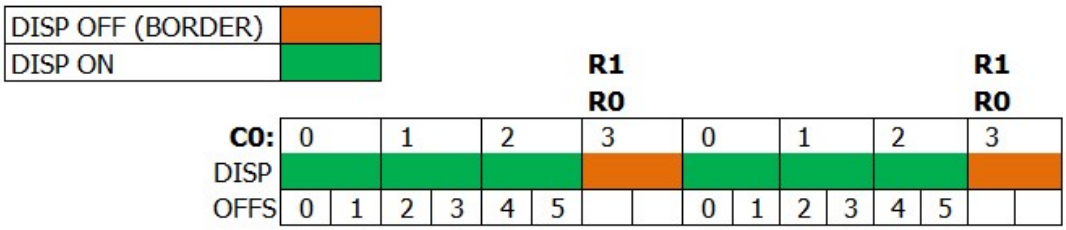
Update of R1 not considered (too late)
Update of R1 ok (just in time)

17.6 BORDER INTERLIGNE

17.6.1 R1=R0 ET C0=R0

CRTC 0, 1, 2, 3, 4

Si R1=R0, tous les CRTC vont afficher 1µs de BORDER sur le dernier caractère (C0=R0).
CRTC-R0=3
CRTC-R1=3
CRTC-R12/R13=0

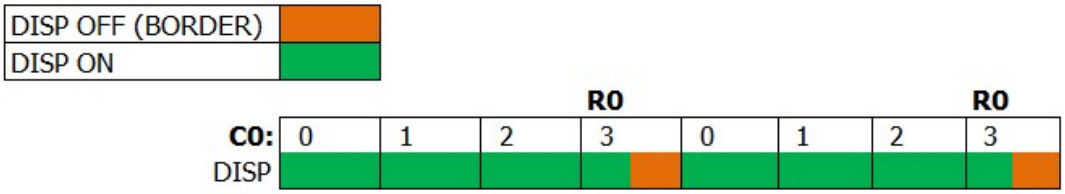


Remarque : Le pointeur en VRAM continuera sur l'offset 6 sur la prochaine "ligne caractère" (lorsque C9 repasse à 0)

17.6.2 R1>R0 ET C0=R0

CRTC 0, 2

La prise en compte du BORDER est anticipée.
1 octet de BORDER est généré (pendant 0.5 µsec) avant que C0 passe à 0.
CRTC-R0=3
CRTC-R1=4
CRTC-R12/R13=0



Remarque 1 : Ce comportement reste vrai quelque soit la valeur de R0.
Si R0=0, alors l'affichage alterne entre 1 octet DISP ON, et 1 octet DISP OFF.
Lorsque C0 atteint R0, les CRTC 0 et 2 génèrent un signal "BORDER" au GATE ARRAY, qui le prend en charge immédiatement. Ces CRTC sont en "avance" sur les caractères affichés par le GATE ARRAY et envoient le signal de BORDER 0.5 µsec trop tôt, qui n'est pas annulé par la condition C0=0.

Remarque 2 : Le pointeur en VRAM restera "coincé" sur VMA'. Si les conditions le permettent (selon le CRTC et l'état des compteurs), VMA' peut être rechargé en mettant à jour R12 et/ou R13.

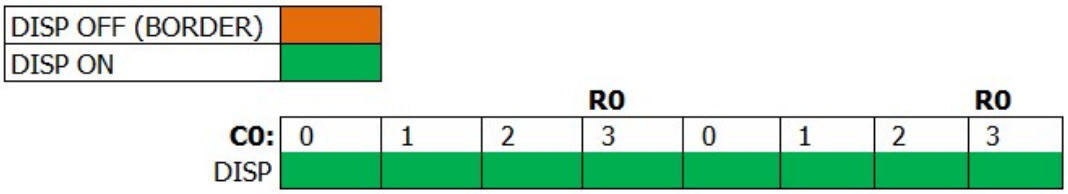
Remarque 3 : Le CRTC 0 dispose d'une fonction permettant de générer des conditions de prise en compte plus tardives du BORDER via son registre R8. Entre la condition C0=R0 et la condition C0=0, la modification des conditions de prise en compte du BORDER permet d'éviter la génération de ce BORDER de 1 octet. Voir chapitre 19.2, page 124.

Remarque 4 : Le CRTC 2 ne dispose pas de la fonction SKEW DISP (R8) et l'octet de BORDER ne peut pas être "annulé" de cette façon. Mais LOGON doit-il vraiment révéler tous ses secrets ?

CRTC 1, 3, 4

CRTC-R0=3
CRTC-R1=4

Sur ces CRTC, aucun octet de BORDER n'est généré entre les "écrans".
Des écrans peuvent donc être créés dans la zone affichable sans qu'un octet apparaisse entre les écrans. Ceci peut permettre de réaliser des plagiats hargneux...



18 AFFICHAGE : REGISTRE R6

18.1 GÉNÉRALITÉS

La fonction de ce registre est de fixer le nombre de lignes-caractères affichées verticalement.
Lorsque ce nombre est atteint, du BORDER est affiché.

Le BORDER est affiché lorsque C4=R6 (1ère ligne-caractère R6).
Cette règle est vraie quelque soit la valeur de C9.

Sauf sur CRTC 3 et 4, la prise en compte de R6 est immédiate sur le C0 courant.

La broche DISPLAY ENABLE du CRTC passe à "OFF" lorsque C4=R6, de la même manière que pour le registre R1 lorsque C0=R1.

En général lorsque la condition est remplie pour afficher du BORDER, il faut attendre un nouvel écran (C4=C9=C0=0) pour rétablir l'affichage des caractères.

Avec le registre R1, le BORDER est désactivé lorsque C0=0 et activé lorsque C0=R1.

Que le border soit généré via R1 ou R6, le pointeur en VRAM continue à être mis à jour.
En l'occurrence, si le BORDER a été activé par la condition sur R6, le pointeur VMA' continue à être mis à jour.

18.2 DÉLAIS ET PRIORITÉS DE BORDER R6

18.2.1 GÉNÉRALITÉS

L'état de la broche DISPLAY ENABLE dépend de 2 groupes de conditions internes du CRTC.

Tant que les "conditions R6" ne sont pas satisfaites, ce sont les "conditions R1" qui définissent l'état de la broche DISPLAY ENABLE.

Lorsque les "conditions R6" sont satisfaites, les "conditions R1" ne sont plus prises en compte.

La condition commune au rétablissement de l'affichage du fond est C4=C9=C0=0.

En général, la condition BORDER R6 est prioritaire sur la condition BORDER R1, mais il y cependant quelques différences selon les CRTC.

18.2.2 CRTC 0, 2

A l'exception de la première ligne d'un écran (C4=C9=0), positionner R6 avec C4 provoque l'activation immédiate et définitive du BORDER jusqu'au prochain écran.

La condition C4=R6 est prise en compte immédiatement (quelque soit la valeur de C0), et BORDER R6 est prioritaire sur BORDER R1.

Ceci est également vrai durant le (ou les) caractères C4 générés pendant un ajustement vertical.

La valeur R6=0 utilisée sur la première ligne (C4=C9=0) est traitée spécifiquement (voir chapitre CONFLITS R6) et il est possible, sous conditions, d'utiliser ce conflit pour **annuler BORDER R6**.

Voir chapitre 18.3.

18.2.3 CRTC 1

Positionner R6 avec C4 provoque l'activation immédiate du BORDER, qui devient définitif jusqu'au prochain écran.

La condition C4=R6 est prise en compte immédiatement (quelque soit la valeur de C0), et BORDER R6 est prioritaire sur BORDER R1.

Comme sur d'autres registres de ce CRTC (par exemple R3) la valeur 0 est prise en compte spécifiquement, et déclenche un BORDER sans que la condition C4=R6 soit requise.

Cependant, si la condition C4=R6 est également remplie lorsque C4=0, alors ce BORDER devient définitif jusqu'à ce que C4=C9=C0=0.

Lorsque R6 est mis à 0 alors que C4<>0, le BORDER est activé tant que R6=0, et il est désactivé dès que R6>0 et que sa nouvelle valeur est différente de C4 (auquel cas, le BORDER R6 est activé pour l'écran).

18.2.4 CRTC 3, 4

Le test de R6 est fait au début de la ligne uniquement.

La mise à jour de R6 **en cours de ligne n'est donc pas prise en compte**.
Si R6 est mis à 0 lorsque C4=0 mais que C0>0, le BORDER n'est pas activé.

Le BORDER est activé uniquement au moment ou C0 passe à 0 lorsque C4=R6.

Ceci est également vrai durant un ajustement vertical.

Remarque : C4 n'est pas incrémenté sur ces 2 CRTC pendant l'ajustement vertical. Cela implique que si R6=R4, alors le BORDER concernera ce caractère ainsi que les lignes d'ajustement vertical définies avec R5.

La valeur de R6=0 n'est pas traitée spécifiquement et ne permet pas d'activer temporairement le BORDER comme sur le CRTC 1.

18.3 CONFLITS R6

18.3.1 GÉNÉRALITÉS

Afin de gérer correctement certains conflits de condition, un traitement particulier est souvent réalisé lorsqu'un registre vaut 0 (notamment sur CRTC 1, le spécialiste du domaine).

C'est notamment le cas lorsque R1=0, car cette valeur satisfait 2 conditions à l'issue contraire (désactivation du BORDER (C0=0), activation du BORDER (C0=R1))
[Dans cette situation, c'est la désactivation du BORDER qui est activée]

Pour R6, la situation a été beaucoup moins bien gérée, selon les CRTC.

18.3.2 CRTC 0, 2

A l'exception de la première ligne d'un écran (C4=C9=0), positionner R6 avec C4 provoque l'activation immédiate et définitive du BORDER jusqu'au prochain écran.

Lorsque C4=R6=0 et C9=0 sur les CRTC 0 et 2, un conflit survient (pour rappel ce conflit n'existe pas lorsque R6>0).

Lorsque R6 est égal à 0 dans cette situation, l'état de DISPLAY ENABLE passe à ON au début du caractère CRTC et repasse à OFF 0.5 µsec après.

Autrement dit, sur la première ligne de l'écran, il y a une alternance d'octets de BORDER et de caractères affichables (le pointeur vidéo continuant à compter normalement).

A chaque caractère CRTC :

- BORDER R6 passe à vrai car on C4=R6 et C9=0
- BORDER R6 repasse à faux car on est sur un nouvel écran (C4=C9=0)

Remarque :

Cette alternance n'a lieu que lorsque la condition R1 est remplie (BORDER R1 est faux) et que les conditions du conflit existent (C4=R6=C9=0).

Lorsque le conflit est annulé (R6 mis à jour avec une valeur > 0), le BORDER ne reste pas activé comme sur les autres lignes (et les autres CRTC).

Dans cette situation cependant, étant donné que R6 a été mis à jour avec 0 au moins 1 fois, le BORDER devient définitif lorsque C0=R1.

Toujours dans cette situation, si on empêche que C0=R1 sur la ligne C4=C9=0 (par exemple R1=R0+1), et que R6 n'est plus égal à 0, alors le BORDER est désactivé sur la ligne suivante.

Autrement dit, **la condition C4=R6=0=BORDER est annulable sur la première ligne.**

Remarque :

La prise en compte sur chaque valeur de C0 de la valeur de R6 permet de cibler précisément des zones ou créer cette alternance BORDER/CARACTERES à l'octet.

Moyennant une jolie rupture ligne à ligne cela permet d'ajouter ponctuellement de jolies couleurs, via du BORDER, au mode graphique 2, par exemple, qu'on appellera le **mode JMLPAJIDA** pour les plus modestes, sinon directement Mode <votre pseudo>.

A noter qu'on peut parvenir à une alternance BORDER/CARACTERE sur CRTC 0 avec R0=0, mais sans que les compteurs C4 et C9 puissent avancer.

18.3.3 CRTC 1

Lorsque R6 est mis à jour avec 0, du BORDER est activé tant que la valeur du registre vaut 0.

La prise en compte sur chaque valeur de C0 de la valeur de R6 permet cibler précisément des zones ou créer ce BORDER.

Cependant, si C4=R6=0 (1ère ligne-caractère d'un nouvel écran) durant cette mise à jour, BORDER R6 devient vrai prioritairement pour tout le reste de l'écran, jusqu'au nouvel écran (C4=C9=C0=0).

18.3.4 CRTC 3, 4

Aucun conflit n'existe puisque la gestion de R6=0 n'existe pas durant la ligne et est testée une seule fois.

Positionner R6 à 0 lorsque C4 et C9 valent 0, mais que C0>0 n'aura aucune incidence avant le nouvel écran (ou le BORDER sera activé).

19 AFFICHAGE : REGISTRE R8

19.1 GÉNÉRALITÉS

Le registre R8 contient des paramètres pour la gestion du mode INTERLACE pour tous les CRTC.

Sur les CRTC 0, 3 et 4, une fonction complémentaire existe, qui permet de retarder la gestion d'activation/désactivation du BORDER, ou désactiver l'affichage comme le fait R6=0 sur CRTC 1 (hors C4=0 sur CRTC 1)).

CRTC	7	6	5	4	3	2	1	0
0	Sc	Sc	Sd	Sd	x	x	i	i
1	x	x	x	x	x	x	i	i
2	x	x	x	x	x	x	i	i
3	x	x	Sd	Sd	x	x	i	i
4	x	x	Sd	Sd	x	x	i	i

Autres CRTC	7	6	5	4	3	2	1	0
MC6845*1	Sc	Sc	Sd	Sd	x	x	i	i
UM6845E	UpdM	US	Sc	Sd	Vdrac	Vdrad	i	i

Interlace		
0	0	No interlace
0	1	Interlace Sync Mode
1	0	No interlace
1	1	Interlace Sync & Video Mode

Skew DISPTMG		
0	0	Non Skew
0	1	One-character skew
1	0	Two-character skew
1	1	Non-output

Skew CUDISP		
0	0	Non Skew
0	1	One-character skew
1	0	Two-character skew
1	1	Non-output

19.2 FONCTIONS « SKEW-DISPTMG »

19.2.1 CRTC 0, 3, 4

Disponible uniquement sur les CRTC 0, 3 et 4, cette fonction permet d'activer le BORDER ou de générer un délai sur la gestion du BORDER R1.

19.2.1.1 BORDER ON : FONCTION 001100xx

Cette fonction permet de désactiver l'affichage des caractères.
Le GATE ARRAY génère du BORDER dans cette situation.
La mise à jour du signal DISPEN (DISPLAY OFF) est immédiatement prise en compte.
Le pointeur de VRAM continue néanmoins d'être incrémenté et le pointeur courant est mis à jour lorsque C0=R1 et C9=C0=0.
Cette fonction n'affecte pas l'état BORDER R6 (lorsque C4=R6) et il est donc possible de basculer sur un des 3 autres états disponibles.

Remarque : Sur CRTC 1, cette affectation directe de DISPEN est possible en mettant R6 à 0. Cependant, si C4=0, alors la condition C4=R6 est remplie et cela provoque la fin d'affichage jusqu'à ce que C4=C9=C0=0. Ce problème n'existe pas avec la fonction BORDER ON.

19.2.1.2 BORDER OFF : FONCTION 000000xx

Cette fonction indique de cesser la gestion des autres fonctions "SKEW".

A tester : positionner la fonction 001100xx , puis attendre que C4=R6 pour activation du BORDER, et repositionner R8 000000xx pour voir si cela peut annuler un BORDER R6.

19.2.1.3 BORDER DELAI 1 & 2: FONCTIONS 000100xx & 001000xx

Ces deux fonctions permettent de gérer un délai sur la gestion du BORDER R1.

Sans que cette fonction soit activée, on a:

- La condition de désactivation du BORDER (début de ligne) est réalisée lorsque C0=0.
- La condition d'activation du BORDER (fin de ligne) est réalisée lorsque C0=R1.

Le délai peut être de 1 ou 2 caractères CRTC (donc 1 ou 2 µsec) selon la fonction.

Lorsque le "délai" est de 1 µsec, la règle pour la gestion du border est:

- Le BORDER est désactivé lorsque C0=1
- Le BORDER est activé lorsque C0=R1+1

Lorsque le "délai" est de 2 µsec, la règle pour la gestion du border est :

- Le BORDER est désactivé lorsque C0=2
- Le BORDER est activé lorsque C0=R1+2

L'affectation du pointeur vidéo, lorsque C9=R9, suit la même logique lorsque C0=R1+1 (ou 2).

Autrement dit, le pointeur vidéo est mémorisé en relation avec la nouvelle position du BORDER.

La prise en compte de ces nouvelles "règles" de test lorsque R8 est modifié est immédiate durant la ligne.

Il existe donc 1 ou 2 µs, selon la fonction utilisée, durant laquelle il est possible "d'annuler" les 2 conditions.

Ceci peut avoir une utilité car l'octet de BORDER généré par les CRTC 0 (et 2) entre "2 lignes" est imputable au CRTC qui va plus vite que le CRTC 1 pour envoyer le signal DISPEN au GATE ARRAY.

Avec le retard que le CRTC 0 peut générer, il est donc possible de supprimer cet octet en annulant les 2 conditions.

19.2.2 CAS PRATIQUE : DÉSINTÉGRATION DU BORDER SUR CRTC 0

On considère que R0=R1=63

	Affichage caractères
	Border

La condition C0=R1+1 est annulée avant C0=63 (Border généré en 63)
La condition C0=1 est annulée avant C0=0 (Caractères affichés en 0)

A

C0:	55	56	**57**	58	59	60	**61**	62	63	0	1
	OUT R8,#10				OUT R8,#0						

B

C0:	56	57	**58**	59	60	61	**62**	63	0	1	2
	OUT R8,#10				OUT R8,#0						

C

C0:	57	58	**59**	60	61	62	**63**	0	1	2	3
	OUT R8,#10				OUT R8,#0						

La condition C0=R1+1 n'a pas pu être annulée avant C0=63 (Pas de Border car "prévu" plus tard)
La condition C0=1 est cependant annulée avant C0=1 (Donc caractères affichés en 0)

Plop! Plus de Border! Y gn'est gn'où ?

D

C0:	58	59	**60**	61	62	63	**0**	1	2	3	4
	OUT R8,#10				OUT R8,#0						

La condition C0=R1+1 n'a pas pu être annulée avant C0=63 (Pas de Border car "prévu" plus tard)
La condition C0=1 n'a cependant pas été annulée avant C0=0 (Donc les caractères sont affichés en C0=1)

E

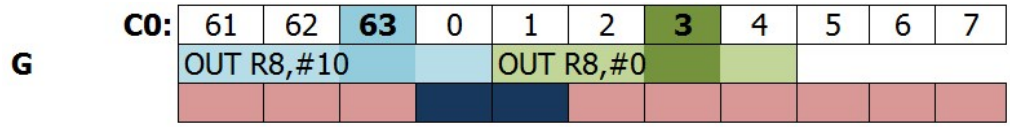
C0:	59	60	**61**	62	63	0	**1**	2	3	4	5
	OUT R8,#10				OUT R8,#0						

F

C0:	60	61	**62**	63	0	1	**2**	3	4	5	6
	OUT R8,#10				OUT R8,#0						

La condition C0=R1+1 n'a pas pu être **activée** avant C0=63 (Border démarre en C0=63)
La condition C0=1 n'a pas été annulée avant C0=0 (Donc les caractères sont affichés en C0=1)

Combo-Border!



Dans le contexte ou R1>R0 (par exemple R0=63, R1=64), alors un octet de BORDER est généré tant que la condition est C0=R1 (R8=000000xx).

Cet octet de BORDER n'est plus affiché lorsque la condition C0=R1+1 est remplie (R8=#10), quelque soit la condition de désactivation du BORDER (C0=0 / C0=1 ou 2).

Ainsi dans l'exemple précédent, pour les cas D, E et F, le BORDER n'est affiché que pour C0=0 (et pas du tout pour le cas C).

19.3 FONCTIONS INTERLACE

19.3.1 GÉNÉRALITÉS

La première chose qu'on peut dire, c'est que la méthode d'approche du sujet par les différents fabricants de CRTC est assez... variée.

Ce premier chapitre concerne le fonctionnement tel qu'il est décrit (très superficiellement) dans les guides techniques des composants. Les chapitres consacrés à cette fonction seront complétés dans des versions plus avancées de ce document (n'ayant pas encore terminé l'analyse de certains résultats pour des raisons de fonds (ah ah)).

Il existe 2 modes interlacés programmables sur tous les CRTC.

L'objectif principal du mode interlacé est de créer un "écran" composé de 2 frame à 50Hz, entrelacés, afin d'afficher une image avec une résolution verticale doublée.

A cette fin, le CRTC doit :

- Etre capable de créer une image composée de 625 lignes distinctes (sur un écran 50Hz)
- Afficher deux frames de 312 lignes, plus une ligne additionnelle appartenant aux 2 frames pour moitié.
  La longueur d'un frame étant alors de 19968 + 32 = 20000 µsec.

Afin d'augmenter la résolution verticale, le CRTC, lors de la VSYNC, retarde le signal de la moitié d'une ligne lorsque C4 passe à R7 pour l'écran affiché avec les lignes paires.

Le canon électron :

- Commence à afficher les données de l'écran pair à partir du coin gauche haut du moniteur.
- Commence à afficher les données de l'écran impair à partir du milieu haut du moniteur.

Le canon à électron, pour les écrans "impairs" parcourt une distance plus courte pour "rejoindre" le bord et commencer à afficher les caractères.

Cet écart de la moitié du temps d'une ligne est sensé permettre au canon d'accéder à l'espace situé entre 2 lignes affichées normalement.

La capacité de gestion de ce type d'image dépend fortement du moniteur :

- D'une part sur sa capacité de persistance des phosphores, pour limiter le scintillement (que certains appelleraient "flashouille" dans certaines contrées froides et humides).
- D'autre part sur la stabilité des éléments haute tension du moniteur pour limiter les distorsions sur les bords de l'écran.

Cependant, le GATE ARRAY fusionne les signaux HSYNC et VSYNC en un signal composite, et "redresse" le signal VSYNC décalé par le CRTC en Interlace.
Du coup, toutes les lignes affichées sur un moniteur CTM sont paires sans faire appel à une autre méthode.

En mode INTERLACE, une image "complète" est composée de 2 "frame" de 50 Hz afin d'afficher une image complète à 25 Hz en 625 lignes (30 Hz en 525 lignes).

Lorsque le premier écran de lignes paires est totalement affiché, une ligne additionnelle est ajoutée. A ce stade, le signal VSYNC a déjà eu lieu avec un retard d'une demi-ligne par rapport à sa position normale en "non interlace".

Cette ligne additionnelle est ajoutée via le gestionnaire interne qui gère l'ajustement vertical.

On peut considérer que cette ligne additionnelle fait partie pour moitié de la fin de l'écran des lignes paires, et pour moitié le début de l'écran des lignes impaires.

19.3.2 LES DEUX MODES INTERLACE

19.3.2.1INTERLACE SYNC MODE : FONCTION 00xx0001

Ce mode est utilisé lorsqu'on souhaite que le CRTC affiche la même information pour l'écran pair et l'écran impair.

L'objectif de ce mode est d'améliorer la qualité des caractères affichés en remplissant les espaces situés entre les lignes en mode "non interlace".

Dans ce mode, le CRTC se contente d'augmenter la résolution en déplaçant la position du signal VSYNC de 1/2 ligne, comme indiqué dans le paragraphe précédent.

L'image dans cette situation utilise la même mémoire vidéo, en affichant deux fois la même chose.
Il n'est en principe pas nécessaire dans ce mode de reprogrammer les registres du CRTC.
La ligne 0 de l'écran des lignes paires est affiché en premier, suivi de la ligne 0 de l'écran des lignes impaires, et ainsi de suite.

Screen:	Even	Odd	Beautiful drawing							
C9=	0									
C9=		0								
C9=	1									
C9=		1								
C9=	2									
C9=		2								
C9=	3									
C9=		3								
C9=	4									
C9=		4								
C9=	5									
C9=		5								
C9=	6									
C9=		6								
C9=	7									
C9=		7								

19.3.2.2 INTERLACE SYNC & VIDEO MODE : FONCTION 00xx0011

Ce mode est qualifié de "Vidéo Mode" car il peut être utilisé pour afficher des images pour lesquelles chacune des 625 lignes est différente.

Contrairement au mode "Interlace Sync Mode", aucune ligne de l'écran construit pour les 2 écrans n'est répétée.

Sur le premier écran, le CRTC affiche les lignes pour lesquelles C9 est pair.
Sur le second écran, le CRTC affiche les lignes pour lesquelles C9 est impair.

A l'issue de l'affichage des 2 écrans (~0.04 seconde), les lignes se suivent dans l'ordre pair/impair.

Etant donné qu'il est nécessaire d'afficher 2 fois plus de lignes, il faut en conséquence programmer les registres du CRTC comme si on construisait un écran de 624 lignes.

Cette logique est mise à l'épreuve par les concepteurs des circuits.

L'ajout de la 625ème ligne est géré automatiquement par le CRTC, pour rappel.

Note: Dans les documents UM6845R dont je dispose, la figure (7) utilisée pour décrire ce mode est erronée (description des numéros de lignes affichées dans les écrans pair/impair).
Cette erreur n'existe pas dans les documents UM6845 de UMC (figure 13), qui est une copie "conforme" du HD6845S de HITACHI (et détecté comme un CRTC 0).

Screen:	Even	Odd	ArtWork from MOMA							
C9=	0									
C9=		1								
C9=	2									
C9=		3								
C9=	4									
C9=		5								
C9=	6									
C9=		7								
C9=	8									
C9=		9								
C9=	10									
C9=		11								
C9=	12									
C9=		13								
C9=	14									
C9=		15								

Le GATE ARRAY, qui fusionne les signaux VSYNC et HSYNC, "redresse" le signal VSYNC, qui démarre donc toujours en haut à gauche.

Etant donné que le CRTC ajoute une ligne "additionnelle" à la fin de l'écran impair, censée être pour moitié sur l'écran pair et moitié sur l'écran impair, cela se traduit par un écran pair qui débute une ligne plus tôt que l'écran impair.

Dans ce contexte, la ligne 0 alterne avec une ligne de BORDER, alors que la ligne 1 alterne avec la ligne 2, la ligne 3 avec la ligne 4, et ainsi de suite...

L'image décrite précédemment apparait ainsi, les pixels "communs" apparaissant alors dans leur couleur d'origine, plutôt qu'en alternance avec l'encre blanche.

C9	Beautiful Drawing Even Screen							
0								
2								
4								
6								
8								
10								
12								
14								

C9	Beautiful Drawing Odd Sceen							
	BORDER or PREVIOUS LINE							
1								
3								
5								
7								
9								
11								
13								
15								

19.3.3 RESTRICTIONS

Il existe des différences de spécification sur les registres à mettre à jour (et leur valeur) en mode Interlace entre les différents circuits. Ces différences sont principalement liées au mode de comptage de C9 et C4 initialement défini par les concepteurs des circuits.

Bien entendu, ces restrictions sont faites pour ne pas être respectées, mais elles donnent cependant des indices intéressants sur le fonctionnement interne des compteurs.

C'est notamment le cas pour le registre R9 sur les CRTC 0, 3 et 4, qui requièrent que R9 soit programmé avec le nombre de ligne d'un caractère vertical, moins 2.
Cela facilite la comparaison de fin des lignes paires, car il suffit d'ignorer le bit 0 pour effectuer la comparaison de fin de caractère et gérer ce bit 0 comme celui de la parité de frame.

L'affichage d'une ligne sur deux provoque l'incrémentation plus rapide de C4 (sur les CRTC 0, 1, 3, 4) et donc de la VSYNC qui survient lorsque C4=R7. La construction d'un écran dans ce mode nécessite d'adapter R4 et R7 à la taille totale de l'écran désiré (sauf pour le CRTC 2, qui a adopté une autre méthodologie).

Lorsqu'un écran de grande taille est ainsi défini, il est possible d'utiliser les "Overscan Bits" ™ pour éviter d'avoir à reprogrammer R12/R13 entre écrans pair et impair, mais je m'égare.

Selon le mode choisi et le CRTC, le nombre de caractères d'une ligne doit être pair (donc R0 doit être impair) (cela a sans doute un rapport avec la gestion de parité du frame).

19.3.4 FONCTION MAL AIMÉE

Le mode INTERLACE est une fonction mal aimée par les développeurs sur CPC.
Note : J'aurais du sortir la démo 4 :-)

Pourtant, cette **fonction est prise en compte en temps réel** et son intérêt n'est pas seulement lié à sa capacité à traiter « convenablement » une image Interlace en produisant, via le GATE ARRAY, un signal utile pour le moniteur.

Dès que la fonction est activée sur une ligne C9 donnée, cela a une incidence sur **le comptage C9 et/ou C4 pour les lignes suivantes**.

Selon la valeur de C9 et sa parité lorsque R8 est modifié, alors la ligne suivante est immédiatement calculée suivant une sauce propre à chaque CRTC.

Autrement dit, dans la perspective de création d'un Interlace « complet », **R8 ne devrait en principe être activé que lorsqu'un écran démarre** (lorsque C4=C9=0).

Pour rappel, la construction d'un écran ne « démarre » pas avec la VSYNC, sauf si R7=0 (cette situation fusionnant alors 2 gestions différentes). Dans le cas contraire, le nombre de lignes du premier écran dépend du moment où le mode Interlace a été activé, et contient alors des lignes paires et impaires jusqu'au moment ou R8 a été modifié (et contient uniquement des lignes paires ou impaires jusqu'à ce que le mode Interlace soit désactivé, afin de revenir à un comptage « classique »).

L'activation du mode Interlace, en plus de modifier le comptage C9 et C4, active deux autres gestions :

- La **gestion d'ajout d'une ligne additionnelle à la fin de l'écran pair**, lorsque C9=C4=0.
- La **gestion d'ajout de 32 µsec avant d'activer la VSYNC**, lorsque C4=R7, selon une logique barbare. A noter que si R7=0, il y « fusion » des deux gestions. Je n'ai pas (encore) tenté d'activer la VSYNC après C0=0 pour vérifier si ces 32 nop représentent un compteur ou une valeur à atteindre.

19.4 PROGRAMMATION VERTICALE EN INTERLACE

19.4.1 CRTC 0

Si N représente le nombre de ligne(s) d'un caractère vertical, alors R9 doit être programmé avec la valeur N-2.

Par exemple si un caractère vertical fait 8 lignes, il faut que R9 contienne 6.
Le CRTC 0 partage cette particularité avec les CRTC 3 et 4.

Si R9 est programmé avec 0, cela signifie qu'il y a 2 lignes minimum affichées (1 par écran 50Hz).

Cependant, SELON LA DOCUMENTATION HITACHI, les choses sont un peu plus compliquées si N est impair (le chapitre ci-dessous est donc à prendre avec une réserve absolue).
Dans cette situation :

- Lorsque C4 est pair :
  - Le CRTC affiche les lignes paires dans l'écran pair
  - Le CRTC affiche les lignes impaires dans l'écran impair
- Lorsque C4 est impair :
  - Le CRTC affiche les lignes impaires dans l'écran pair
  - Le CRTC affiche les lignes paires dans l'écran impair

Le registre R6 doit être programmé comme habituellement en mode non Interlace.

R7 doit être programmé en relation avec R4 pour l'écran "total" à 25 Hz.
R4 doit être programmé avec le nombre de caractères verticaux moins 2.

19.4.2 CRTC 1

N représente le nombre de ligne(s) d'un caractère vertical, alors R9 doit être programmé avec la valeur N-1.

R4 et R7 doivent être programmés avec le nombre de caractères de l'écran "total" à 25 Hz.

19.4.3 CRTC 2

Si N représente le nombre de ligne(s) d'un caractère vertical, alors R9 doit être programmé avec la valeur N-1.

Le nombre de ligne(s) d'un caractère vertical doit cependant être pair (R9 doit être programmé avec une valeur impaire).

R6 doit également être pair, et doit être divisé par 2 par rapport à la valeur qu'il devrait avoir normalement.

R4 et R7 n’ont pas besoin d’être reprogrammés.

A priori, ce CRTC gère son compteur C4 grâce à la valeur de son premier compteur C9.
(Nous allons voir plus loin que ce CRTC dispose, selon moi, de 2 compteurs C9).

Il existe donc des situations ou C4 n’est pas incrémenté lorsque C9=R9 mais le pointeur vidéo est tout de même mis à jour. Ainsi R6 concerne 2 caractères et non un seul, et c’est pourquoi il est « indiqué » dans les documentations techniques que R6 doit être pair.

19.5 COMPTAGES EN INTERLACE VIDEOMODE

19.5.1 CRTC 0

Lorsque le mode IVM est activé, le CRTC considère C9 comme décalé à gauche de 1 bit.
Cela revient à considérer que la nouvelle valeur de C9 est l'ancienne multipliée par 2.
Le bit 0 contient alors la parité, et participe à la gestion paire ou impaire de C9.

La gestion d'incrémentation de C9 continue à être traitée sur la valeur « non multipliée ».
Chaque incrémentation correspond alors à un ajout de 2 au compteur.

Cependant, c'est bien la valeur de C9 « multipliée » qui est comparée à R9 pour gérer la remise à 0 de C9 et gérer C4.

On peut formuler ça ainsi lorsque R8=3 (ou 1) :

Si (C9x2)=R9
Alors
    C9=0 ; Gestion C4 (C4++ ou C4=0)
Sinon
    C9++
Fin Si
C9Disp=(C9x2) or Parité

La valeur de C9 lorsque R8 passe à 3 peut conduire à un débordement de C9.
Par exemple, si R9 vaut 6, et C9=3 lorsque R8 est mis à jour avec 3, alors le C9 suivant sera (C9+1)x2, soit 8 (ou 9 si la parité est impaire). C9 est alors supérieur à R9 et il va alors continuer à s'incrémenter de 2 en 2 jusqu'à ce qu'il soit de nouveau égal à 6 (après avoir atteint sa valeur limite et rebouclé).

Lorsque R8 revient à 0, C9 est divisé par 2 et la gestion de parité cesse.
Par exemple, si R9 vaut 6, et C9=4 lorsque R8 est mis à jour avec 0, alors le C9 suivant sera (C9/2)+1, soit 3.

Les schémas pages suivantes décrivent différentes situations de comptage, lors du passage en mode IVM ou lors de la sortie du mode IVM.

Remarque : La période IVM de test couvre seulement quelques lignes de l'écran, et ce après la VSYNC, avant cette dernière et avant que C4 atteigne R6. R8 vaut 0 pendant la VSYNC.

Les valeurs en rouge n'ont pas (encore) été vérifiées.
Je les indique car la probabilité est assez élevée (ce sujet est vaste...).

R9=6 et R8=0 avant de passer à 3 (ou repasser à 0).

Passage en mode IVM sur CRTC 0 :

EVEN FRAME

C4	C9	R8 UPDATE
0	0	**R8=3**
0	2	
0	4	
0	6	
1	0	
1	2	
1	4	
1	6	

ODD FRAME

C4	C9	R8 UPDATE
0	0	**R8=3**
0	3	
0	5	
0	7	
1	1	
1	3	
1	5	
1	7	

C4	C9	R8 UPDATE
0	0	
0	1	**R8=3**
0	4	
0	6	
1	0	
1	2	
1	4	
1	6	

C4	C9	R8 UPDATE
0	0	
0	1	**R8=3**
0	5	
0	7	
1	1	
1	3	
1	5	
1	7	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	**R8=3**
0	6	
1	0	
1	2	
1	4	
1	6	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	**R8=3**
0	7	
1	1	
1	3	
1	5	
1	7	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	
0	3	**R8=3**
0	8	
0	10	
0	12	
0	14	
0	16	
0	18	
0	20	
0	22	
0	24	
0	26	
0	28	
0	30	
0	0	
0	2	
0	4	
0	6	
1	0	
1	2	
1	4	
1	6	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	
0	3	**R8=3**
0	9	
0	11	
0	13	
0	15	
0	17	
0	19	
0	21	
0	23	
0	25	
0	27	
0	29	
0	31	
0	1	
0	3	
0	5	
0	7	
1	1	
1	3	
1	5	
1	7	

EVEN FRAME

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	
0	3	
0	4	**R8=3**
0	10	
0	12	
0	14	
0	16	
0	18	
0	20	
0	22	
0	24	
0	26	
0	28	
0	30	
0	0	
0	2	
0	4	
0	6	
1	0	
1	2	
1	4	
1	6	

ODD FRAME

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	
0	3	
0	4	**R8=3**
0	11	
0	13	
0	15	
0	17	
0	19	
0	21	
0	23	
0	25	
0	27	
0	29	
0	31	
0	1	
0	3	
0	5	
0	7	
1	1	
1	3	
1	5	
1	7	

Sortie du mode IVM sur CRTC 0 :

EXIT IVM MODE

C4	C9	R8 UPDATE
0	0	**R8=3**
0	2	
0	4	
0	6	
1	0	**R8=0**
1	1	
1	2	
1	3	
1	4	
1	5	
1	6	
1	7	

C4	C9	R8 UPDATE
0	0	**R8=3**
0	2	
0	4	
0	6	
1	0	
1	2	**R8=0**
1	1	
1	2	
1	3	
1	4	
1	5	
1	6	

C4	C9	R8 UPDATE
0	0	**R8=3**
0	2	
0	4	
0	6	
1	0	
1	2	
1	4	**R8=0**
1	3	
1	4	
1	5	
1	6	
1	7	

C4	C9	R8 UPDATE
0	0	**R8=3**
0	2	
0	4	
0	6	
1	0	
1	2	
1	4	
1	6	**R8=0**
1	4	
1	5	
1	6	
1	7	

C4	C9	R8 UPDATE
0	1	**R8=3**
0	3	
0	5	
0	7	
1	1	**R8=0**
1	1	
1	2	
1	3	
1	4	
1	5	
1	6	
1	7	

C4	C9	R8 UPDATE
0	1	**R8=3**
0	3	
0	5	
0	7	
1	1	
1	3	**R8=0**
1	2	
1	3	
1	4	
1	5	
1	6	
1	7	

C4	C9	R8 UPDATE
0	1	**R8=3**
0	3	
0	5	
0	7	
1	1	
1	3	
1	5	**R8=0**
1	3	
1	4	
1	5	
1	6	
1	7	

C4	C9	R8 UPDATE
0	1	**R8=3**
0	3	
0	5	
0	7	
1	1	
1	3	
1	5	
1	7	**R8=0**
2	0	
2	1	
2	2	
2	3	

19.5.2 CRTC 1

Le calcul de C9 est orienté sur la conservation de la parité.
Lorsque le mode IVM est activé sur un C9 donné, le C9 suivant suit la formule suivante :
C9-suivant=((C9-courant and %11110) + 2) or Parite

Lors de la comparaison C9/R9, le bit 0 n'est pas pris en compte lorsqu'il est question de gérer C4 (incrémenté ou remis à 0 si C9=R9).

Cependant, C9 ne repasse à 0 que si C9=R9.
Ainsi **C4 peut s'incrémenter sans que C9 revienne à 0**.

Par exemple, si R9=7, C9=6 et R8 passe de 3 à 0.
Ligne N+1 : C9 = R9 and %11110 >> C4=C4+1, C9=C9+1.
Ligne N+2 : C9 = R9 (IVM Off) >> C4=C4+1, C9=0

A partir de mes observations actuelles, la parité semble armée la première fois que R8=3. Si C9 est impair lorsque R8=3, alors la prochaine activation de R8=3 provoquera l'inversion de parité.

Les schémas pages suivantes décrivent différentes situations de comptage, lors du passage en mode IVM ou lors de la sortie du mode IVM.

Remarque : La période IVM de test couvre seulement quelques lignes de l'écran, et ce après la VSYNC, avant cette dernière et avant que C4 atteigne R6. R8 vaut 0 pendant la VSYNC.

Les valeurs en rouge n'ont pas (encore) été vérifiées.
Je les indique car la probabilité est assez élevée (ce sujet est vaste...).

R9=7 et R8=0 avant de passer à 3 (ou repasser à 0).

Passage en mode IVM sur CRTC 1 :

ENTER IVM MODE ON EVEN C9

C4	C9	R8 UPDATE
0	0	**R8=3**
0	2	
0	4	
0	6	
1	0	
1	2	
1	4	
1	6	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	**R8=3**
0	4	
0	6	
1	0	
1	2	
1	4	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	
0	3	
0	4	**R8=3**
0	6	
1	0	
1	2	
1	4	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	
0	3	
0	4	
0	5	
0	6	**R8=3**
1	0	
1	1	

ENTER IVM MODE ON ODD C9

1ST FRAME

C4	C9	R8 UPDATE
0	0	
0	**1**	**R8=3**
0	3	
0	5	
0	7	
1	1	
1	3	
1	5	
1	7	

2ND FRAME

C4	C9	R8 UPDATE
0	0	
0	**1**	**R8=3**
0	2	
0	4	
0	6	
1	0	
1	2	
1	4	
1	6	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	
0	**3**	**R8=3**
0	5	
0	7	
1	1	
1	3	
1	5	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	
0	**3**	**R8=3**
0	4	
0	6	
1	0	
1	2	
1	4	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	
0	3	
0	4	
0	**5**	**R8=3**
0	7	
1	1	
1	3	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	
0	3	
0	4	
0	**5**	**R8=3**
1	6	
1	0	
1	2	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	
0	3	
0	4	
0	5	
0	6	
0	7	**R8=3**
1	1	
1	3	
1	5	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	
0	3	
0	4	
0	5	
1	6	
1	7	**R8=3**
1	0	
1	2	
1	4	

Sortie du mode IVM sur CRTC 1 :

EXIT IVM MODE, EVEN C9

C4	C9	R8 UPDATE
0	0	**R8=3**
0	2	
0	4	
0	6	
1	0	**R8=0**
1	1	
1	2	
1	3	
1	4	
1	5	
1	6	
1	7	

C4	C9	R8 UPDATE
0	0	**R8=3**
0	2	
0	4	
0	6	
1	0	
1	2	**R8=0**
1	3	
1	4	
1	5	
1	6	
1	7	
2	0	

C4	C9	R8 UPDATE
0	0	**R8=3**
0	2	
0	4	
0	6	
1	0	
1	2	
1	4	**R8=0**
1	5	
1	6	
1	7	
2	0	
2	1	

C4	C9	R8 UPDATE
0	0	**R8=3**
0	2	
0	4	
0	6	
1	0	
1	2	
1	4	
1	6	**R8=0**
2	7	
3	0	
3	1	
3	2	

EXIT IVM MODE, ODD C9

C4	C9	R8 UPDATE
0	1	**R8=3**
0	3	
0	5	
0	7	
1	1	**R8=0**
1	2	
1	3	
1	4	
1	5	
1	6	
1	7	
2	0	

C4	C9	R8 UPDATE
0	1	**R8=3**
0	3	
0	5	
0	7	
1	1	
1	3	**R8=0**
1	4	
1	5	
1	6	
1	7	
2	0	
2	1	

C4	C9	R8 UPDATE
0	1	**R8=3**
0	3	
0	5	
0	7	
1	1	
1	3	
1	5	**R8=0**
1	6	
1	7	
2	0	
2	1	
2	2	

C4	C9	R8 UPDATE
0	1	**R8=3**
0	3	
0	5	
0	7	
1	1	
1	3	
1	5	
1	7	**R8=0**
2	0	
2	1	
2	2	
2	3	

19.5.3 CRTC 2

Il n'est pas nécessaire de reprogrammer R4 afin que l'écran soit constitué de 312 lignes (lorsque R4=38 et R9=7). C'est également le cas lorsque le mode IVM est programmé sur une fraction verticale de l'écran. C4 n'est pas faussé lorsque R8 est mis à jour, même temporairement.

R7 n'a également pas besoin d'être reprogrammé pour que l'écran soit synchronisé.

Cela implique deux choses importantes :

- C4 ne suit pas la nouvelle logique de comptage de C9, et donc la mise à jour de VMA' avec VMA a lieu lorsque C0=R1 indépendamment de C4.
- Il existe deux compteurs C9 distincts gérant R9 dans le circuit afin d'éviter qu'un décalage de C4 puisse se produire.

Comme sur d'autres CRTC, la valeur limite de C9 est traitée en excluant le bit 0 de R9 pour le test, et la parité du frame est ajoutée ensuite pour obtenir la valeur finale de C9.

Si on nomme le compteur « standard » C9a, et celui prévu pour l'Interlace C9b, chacun de ces compteurs gère le dépassement R9. Ainsi, le passage de R8=3 consiste à utiliser C9b, et le passage de R8=0 consiste à utiliser C9a.

								R9
C9=C9a	0	1	2	3	4	5	6	**7**
C9=C9b Even Frame	0	2	4	6	0	2	4	6
C9=C9b+1 Odd Frame	1	3	5	7	1	3	5	7

Si R8 passe de 0 à 3 lorsque C9 vaut 5, alors C9 passe à 2 sur un frame pair (3 frame impair).
Si R8 passe de 3 à 0 lorsque C9 vaut 5, alors C9 passe à 2 ou 6 selon sa position.

A priori, cette translation est immédiatement prise en compte dans la constitution de l'adresse affichée, et ce y compris durant la ligne, à partir de la position C0 ou R8 est modifié.

On peut supposer que C4 compte 1 fois sur 2 lorsque R8=3 ou que C4 compte en se basant uniquement sur la valeur de C9a. En application du principe de parcimonie (Guillaume d'Ockham), la solution la plus simple est de considérer que C4 compte avec C9a uniquement.

Le BORDER est activé lorsque C4=R6. Les caractères affichés ne sont plus « associés » à C4 et une valeur du compteur peut alors correspondre à plusieurs caractères. Un caractère étant ici considéré comme la mise à jour de CRTC-VMA' lorsque C9b=R9 and %11110 et C0=R1.

Les schémas pages suivantes décrivent différentes situations de comptage, lors du passage en mode IVM ou lors de la sortie du mode IVM.

Remarque : La période IVM de test couvre seulement quelques lignes de l'écran, et ce après la VSYNC, avant cette dernière et avant que C4 atteigne R6. R8 vaut 0 pendant la VSYNC.

Les valeurs en rouge n'ont pas (encore) été vérifiées.
Je les indique car la probabilité est assez élevée (ce sujet est vaste...).

R9=7 et R8=0 avant de passer à 3 (ou repasser à 0).

Passage en mode IVM sur CRTC 2 :

EVEN FRAME

C4	C9	R8 UPDATE
0	0	**R8=3**
0	2	
0	4	
0	6	
0	0	
0	2	
0	4	
0	6	

C4	C9	R8 UPDATE
0	0	
0	1	**R8=3**
0	4	
0	6	
0	0	
0	2	
0	4	
0	6	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	**R8=3**
0	6	
0	0	
0	2	
0	4	
0	6	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	
0	3	**R8=3**
0	0	
0	2	
0	4	
0	6	

ODD FRAME

C4	C9	R8 UPDATE
0	0	**R8=3**
0	3	
0	5	
0	7	
0	1	
0	3	
0	5	
0	7	

C4	C9	R8 UPDATE
0	0	
0	1	**R8=3**
0	5	
0	7	
0	1	
0	3	
0	5	
0	7	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	**R8=3**
0	7	
0	1	
0	3	
0	5	
0	7	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	
0	3	**R8=3**
0	1	
0	3	
0	5	
0	7	

Sortie du mode IVM sur CRTC 2 :

EXIT IVM MODE

EVEN FRAME

C4	C9	R8 UPDATE
0	6	
1	0	**R8=3**
1	2	
1	4	
1	6	
2	0	**R8=0**
2	1	
2	2	
2	3	
2	4	
2	5	
2	6	
2	7	

ODD FRAME

C4	C9	R8 UPDATE
0	7	
1	1	**R8=3**
1	3	
1	5	
1	7	
2	1	**R8=0**
2	1	
2	2	
2	3	
2	4	
2	5	
2	6	
2	7	

C4	C9	R8 UPDATE
0	6	
1	0	**R8=3**
1	2	
1	4	
1	6	
2	0	
2	2	**R8=0**
2	2	
2	3	
2	4	
2	5	
2	6	
2	7	

C4	C9	R8 UPDATE
0	7	
1	1	**R8=3**
1	3	
1	5	
1	7	
2	1	
2	3	**R8=0**
2	2	
2	3	
2	4	
2	5	
2	6	
2	7	

C4	C9	R8 UPDATE
0	6	
1	0	**R8=3**
1	2	
1	4	
1	6	
2	0	
2	2	
2	4	**R8=0**
2	3	
2	4	
2	5	
2	6	
2	7	

C4	C9	R8 UPDATE
0	7	
1	1	**R8=3**
1	3	
1	5	
1	7	
2	1	
2	3	
2	5	**R8=0**
2	3	
2	4	
2	5	
2	6	
2	7	

EXIT IVM MODE

EVEN FRAME

C4	C9	R8 UPDATE
0	6	
1	0	**R8=3**
1	2	
1	4	
1	6	
2	0	
2	2	
2	4	
2	6	**R8=0**
2	4	
2	5	
2	6	
2	7	

ODD FRAME

C4	C9	R8 UPDATE
0	7	
1	1	**R8=3**
1	3	
1	5	
1	7	
2	1	
2	3	
2	5	
2	7	**R8=0**
2	4	
2	5	
2	6	
2	7	

19.5.4 CRTC 3, 4

Il n'existe aucune documentation sur l'implémentation de l'Interlace sur ces CRTC émulés, mais la logique est cependant assez simple.

Il est important de se rappeler que lorsque R9 est mis à jour avec une valeur inférieure à C9, alors C9 passe à 0.

Comme sur les CRTC 0 et 1, l'activation du mode IVM modifie le comptage de C4, et nécessite donc d'adapter les valeurs de R4, R6 et R7 selon les évolutions de C4.

Pour que le mode IVM suive une logique de lignes paires/impaires, il est cependant nécessaire de programmer R9 **de la même façon que sur un CRTC 0**.

Il faut donc que R9 contienne le nombre de lignes d'un caractère moins 2. Soit 6 si un caractère est composé de 8 lignes. Dans le cas contraire la gestion de C9 entraine un changement de parité lorsque C9 revient à 0.

Dès que R8 vaut 3, ce CRTC exécute deux fois la gestion C9 suivante (1 seule fois sinon) :

Si C9 > R9
Alors
    C9=0 ; Gestion C4 (Si C4=R4, C4=0, sinon C4++)
Sinon
    C9++
Fin Si

Exemple : Contexte R9=7 et C9=6 (ligne N)

Ligne N+1	1ère itération : C9 est incrémenté une première fois : C9=7
	2ème itération : C9 est incrémenté une seconde fois : C9=8
Ligne N+2	1ère itération : C9 est supérieur à 7. Il passe à 0. C9=0 et C4=C4+1
	2ème itération : C9 est incrémenté une seconde fois : C9=1

Dans le contexte de mes tests, je n'ai pas remarqué l'activation d'une parité d'un frame sur l'autre dès lors que R8 est resté à 3 dans les conditions décrites ci-dessous.

Les schémas pages suivantes décrivent différentes situations de comptage, lors du passage en mode IVM ou lors de la sortie du mode IVM.

Remarque : La période IVM de test couvre seulement quelques lignes de l'écran, et ce après la VSYNC, avant cette dernière et avant que C4 atteigne R6. R8 vaut 0 pendant la VSYNC.

Les valeurs en rouge n'ont pas (encore) été vérifiées.
Je les indique car la probabilité est assez élevée (ce sujet est vaste...).

R9=6 ou R9=7, et R8=0 avant de passer à 3 (ou repasser à 0).

Passage en mode IVM sur CRTC 3 & 4 :

R9=7

C4	C9	R8 UPDATE
0	0	**R8=3**
0	2	
0	4	
0	6	
0	8	
1	1	
1	3	
1	5	
1	7	
2	0	
2	2	
2	4	
2	6	
2	8	

R9=6

C4	C9	R8 UPDATE
0	0	**R8=3**
0	2	
0	4	
0	6	
1	0	
1	2	
1	4	
1	6	
2	0	
2	2	
2	4	
2	6	
3	0	
3	2	

C4	C9	R8 UPDATE
0	0	
0	1	**R8=3**
0	3	
0	5	
0	7	
1	0	
1	2	
1	4	
1	6	
1	8	
2	1	
2	3	
2	5	
2	7	

C4	C9	R8 UPDATE
0	0	
0	1	**R8=3**
0	3	
0	5	
0	7	
1	1	
1	3	
1	5	
1	7	
2	1	
2	3	
2	5	
2	7	
2	1	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	**R8=3**
0	4	
0	6	
0	8	
1	1	
1	3	
1	5	
1	7	
2	0	
2	2	
2	4	
2	6	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	**R8=3**
0	4	
0	6	
0	8	
1	0	
1	2	
1	4	
1	6	
2	0	
2	2	
2	4	
2	6	

R9=7

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	
0	3	**R8=3**
0	5	
0	7	
1	0	
1	2	
1	4	
1	6	
1	8	
2	1	
2	3	
2	5	

R9=6

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	
0	4	**R8=3**
0	6	
1	0	
1	2	
1	4	
1	6	
2	0	
2	2	
2	4	
2	6	
3	0	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	
0	3	
0	4	**R8=3**
0	6	
0	8	
1	1	
1	3	
1	5	
1	7	
2	0	
2	2	
2	4	

C4	C9	R8 UPDATE
0	0	
0	1	
0	2	
0	3	
0	4	**R8=3**
0	6	
1	0	
1	2	
1	4	
1	6	
2	0	
2	2	
2	4	
2	6	

Sortie du mode IVM sur CRTC 3 & 4 :

EXIT IVM MODE

R9=7

C4	C9	R8 UPDATE
0	0	**R8=3**
0	2	
0	4	
0	6	
0	8	
1	1	**R8=0**
1	2	
1	3	
1	4	
1	5	
1	6	
1	7	
2	0	

R9=6

C4	C9	R8 UPDATE
0	1	**R8=3**
0	3	
0	5	
0	7	
1	1	**R8=0**
1	2	
1	3	
1	4	
1	5	
1	6	
1	7	
2	0	
2	1	

C4	C9	R8 UPDATE
0	8	**R8=3**
1	1	
1	3	
1	5	
1	7	
2	0	**R8=0**
2	1	
2	2	
2	3	
2	4	
2	5	
2	6	
2	7	

C4	C9	R8 UPDATE
0	7	**R8=3**
1	1	
1	3	
1	5	
1	7	
2	1	**R8=0**
2	2	
2	3	
2	4	
2	5	
2	6	
2	7	
3	0	

C4	C9	R8 UPDATE
0	7	**R8=3**
1	0	
1	2	
1	4	
1	6	
1	8	**R8=0**
2	0	
2	1	
2	2	
2	3	
2	4	
2	5	
2	6	

R9=7

C4	C9	R8 UPDATE
0	5	**R8=3**
0	7	
1	0	
1	2	
1	4	
1	6	**R8=0**
1	7	
2	0	
2	1	
2	2	
2	3	
2	4	
2	5	

R9=6

C4	C9	R8 UPDATE
0	4	**R8=3**
0	6	
1	0	
1	2	
1	4	
1	6	**R8=0**
1	7	
2	0	
2	1	
2	2	
2	3	
2	4	
2	5	

C4	C9	R8 UPDATE
0	2	**R8=3**
0	4	
0	6	
0	8	
1	1	
1	3	**R8=0**
1	4	
1	5	
1	6	
1	7	
2	0	
2	1	
2	2	

C4	C9	R8 UPDATE
0	1	**R8=3**
0	3	
0	5	
0	7	
1	1	
1	3	**R8=0**
1	4	
1	5	
1	6	
1	7	
2	0	
2	1	
2	2	

C4	C9	R8 UPDATE
0	4	**R8=3**
0	6	
0	8	
1	1	
1	3	
1	5	**R8=0**
1	6	
1	7	
2	0	
2	1	
2	2	
2	3	
2	4	

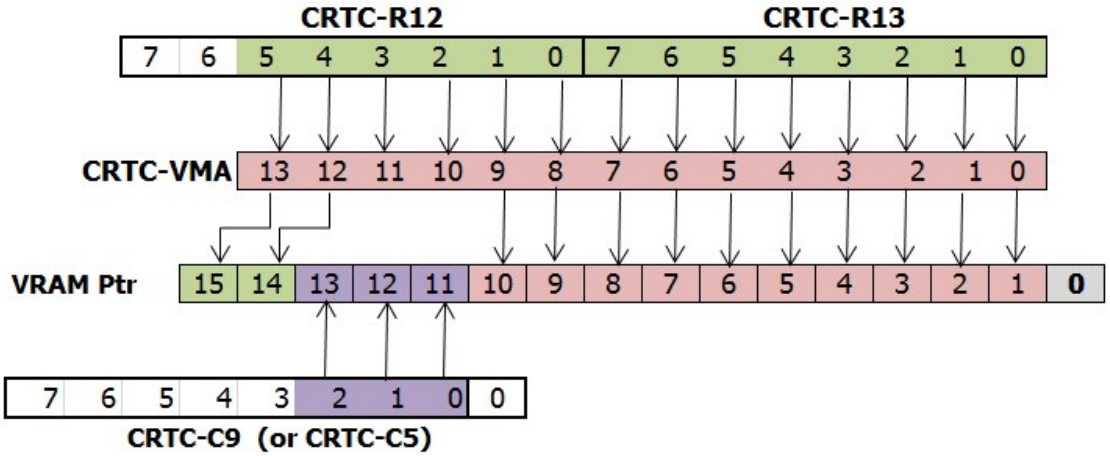
C4	C9	R8 UPDATE
0	3	**R8=3**
0	5	
0	7	
1	1	
1	3	
1	5	**R8=0**
1	6	
1	7	
2	0	
2	1	
2	2	
2	3	
2	4	

20 POINTEUR VIDEO:REGISTRES R12/R13

20.1 GÉNÉRALITÉS

Ces deux registres permettent de définir, en conjonction avec C9 (ou C5), l'adresse mémoire de base communiquée par le CRTC au GATE ARRAY pour qu'il affiche ses caractères.

20.2 CALCUL DU POINTEUR VIDÉO



Bit 0	**Always at 0 because the CRTC works in words**
Bits 1 to 10	**From bits 0 to 9 of CRTC-VMA**
Bits 11 to 13	**From bits 0 to 2 of C9 (or C5 when there is a vertical adjustment with CRTC 3 and 4)**
Bits 14 and 15	**From bits 12 and 13 of CRTC-VMA**

Lorsque les conditions de mise à jour de VMA/VMA' sont remplies, alors VMA ou VMA'=R12/R13. Voir chapitre 14, page 84.

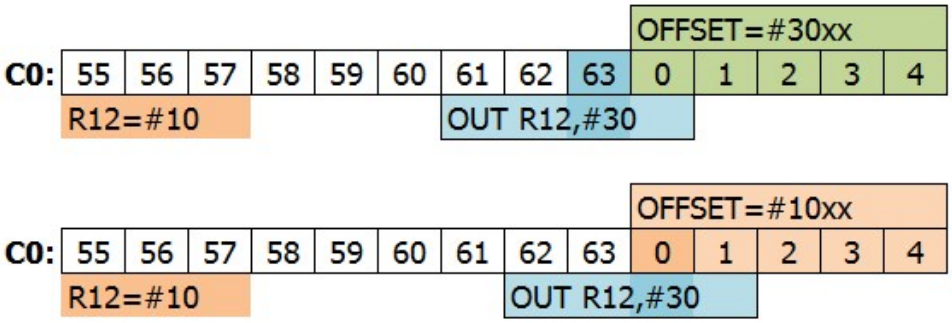
20.3 CONDITIONS DE MISE A JOUR

20.3.1 CRTC 0

Lorsque les compteurs C4, C9 et C0 passent à 0, les pointeurs vidéo (VMA' & VMA) sont initialisés avec R12/R13.

A noter que si les "écrans" font 2 µsec (R4=R9=0 et R0=1) alors C4 passe par les valeurs 0 et 1 alternativement, permettant une prise en compte de R12 et/ou R13 toutes les 4 µs.
(Il s'agit du déclenchement de la gestion "R5" de ligne additionnelle, qui génère une ligne malgré R5=0, et incrémente donc C4 malgré R4=0).

La mise à jour de R12 et R13 est prise en compte immédiatement.



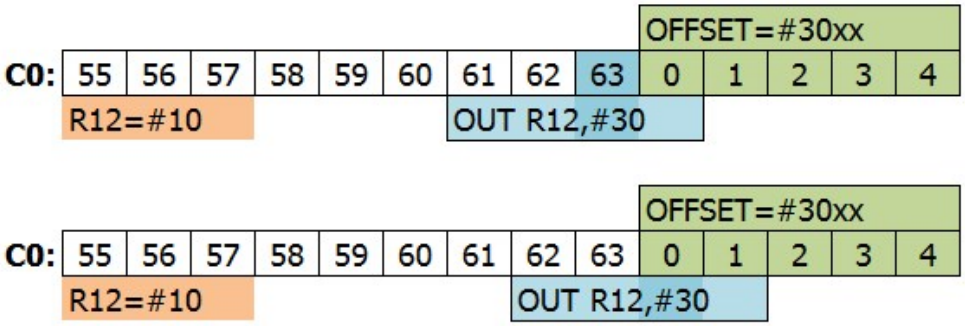
Le CRTC 0 charge **VMA' & VMA avec R12/R13 si C4=0 et C0=0.**

20.3.2 CRTC 1

Lorsque C0 et C9 passent à 0 et que C4=0, le pointeur vidéo VMA est initialisé avec R12/R13.

A noter que la mise à jour de R12 et R13 est immédiate.

Si les conditions sont réunies, il est possible de changer l'offset sur des écrans de 1 µs (R0=0).



Le CRTC 1 charge **VMA avec R12/R13 tant que C4=0.**

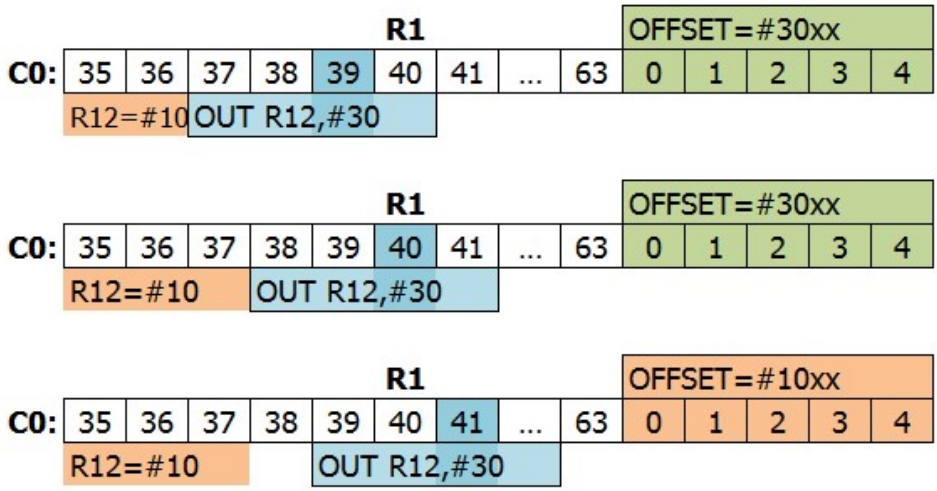
20.3.3 CRTC 2

Lorsque les compteurs C4, C9 et C0 passent à 0, le pointeur VMA' est initialisé avec R12/R13.

Contrairement aux autres CRTC, c'est VMA' qui est mis à jour avec R12/R13.

Pour rappel VMA' est le pointeur transitoire mis à jour lorsque C0 atteint R1, et qui vient mettre à jour VMA lorsque C9=R9.

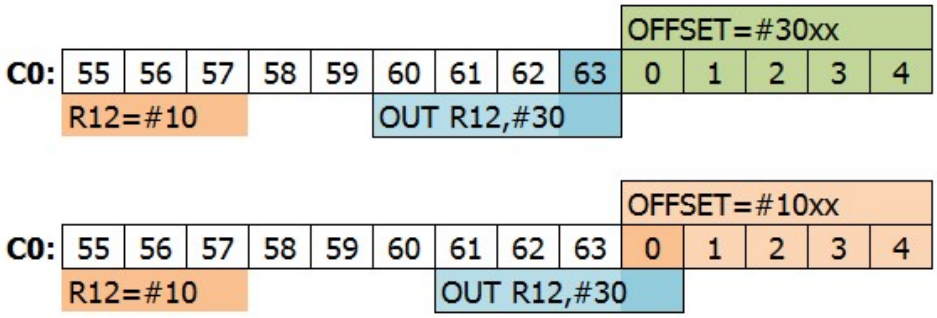
Si la mise à jour des registres R12/R13 est immédiate, la prise en compte pour la nouvelle ligne ne peut plus avoir lieu lorsque C0 dépasse R1.



Le CRTC 2 accepte de charger VMA' avec R12/R13 si C4=0 et C0=0.

20.3.4 CRTC 3 & 4

Lorsque les compteurs C4, C9 et C0 passent à 0, les pointeurs vidéo (VMA' & VMA) sont initialisés avec R12/R13.



Les CRTC 3 et 4 acceptent de charger VMA' & VMA avec R12/R13 si C4=0 et C0=0.

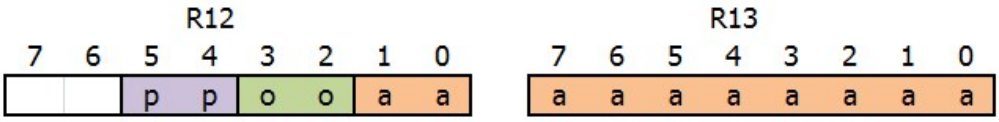
20.4 DELAIS DE PRISE EN COMPTE

Voir le chapitre sur le registre R0 pour le détail précis de la mise à jour du pointeur vidéo.

20.5 OVERSCAN-BITS ;-)

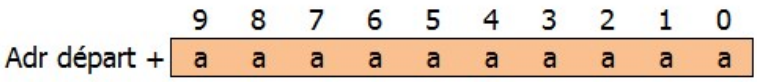
La construction de l'adresse mémoire est particulière dans la mesure ou :

- Les bits de C9/C5 sont "inamovibles" dans le pointeur final.
- Le compteur interne d'adresse compte sur 14 bits avec les bits 10.11.12 qui ne "servent pas" (du moins en apparence).



p	p	Adr départ
0	0	&0000
0	1	&4000
1	0	&8000
1	1	&C000

o	o	Taille gérée
0	0	16k
0	1	16k
1	0	16k
1	1	32k



Lorsque R12 et R13 sont transférés dans VMA/VMA', c'est VMA qui sert à afficher les caractères.

Bien que les bits 11, 12 et 13 du pointeur définitif proviennent de C9 (ou C5), le compteur VMA compte sur 14 bits. Aussi, lorsque le comptage arrive au bout de 10 bits d'adresse, les bits 2 et 3 venus de R12 participent au comptage, même si il ne servent pas directement pour le pointeur vidéo.

Si ces bits sont tous les deux à 1, cela provoque un report sur les bits 4 et 5 de R12.
Ces 2 bits représentent les bits 14 et 15 du pointeur vidéo.

La conséquence est donc un changement de page lorsqu'on arrive au bout des 10 bits.

Il est donc possible d'afficher une image de plus de 16k sans avoir recours à une rupture avec cette méthode. Certains pourraient refuser aux bit 2 et 3 de R12 le droit légitime d'être appelés "Overscan Bits ™ ", mais ce serait du pur révisionnisme spéculatif...

21 FULLSCREEN & CENTRAGE

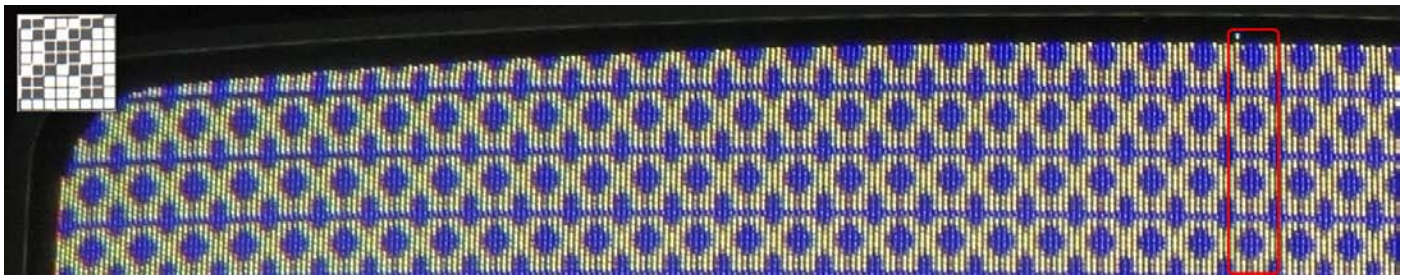
21.1 PRÉAMBULE

Les écrans cathodiques du CPC sont convexes avec une courbure d'écran assez visible.
A cause de cette forme et de l'angle de déviation du faisceau d'électrons, les pixels situés sur les bords sont plus grands que ceux situés au centre.

De plus, le boitier qui entoure le tube cathodique dispose de coins et bords arrondis qui épousent « à peu près » la forme du tube cathodique, mais masquent cependant au passage quelques pixels complémentaires. Les données indiquées dans ce chapitre correspondent au cas général. A cause de la nature analogique des écrans, il peut y avoir quelques petites variations de taille ou de positionnement entre 2 écrans. N'oublions que le CPC a plus de 30 ans.

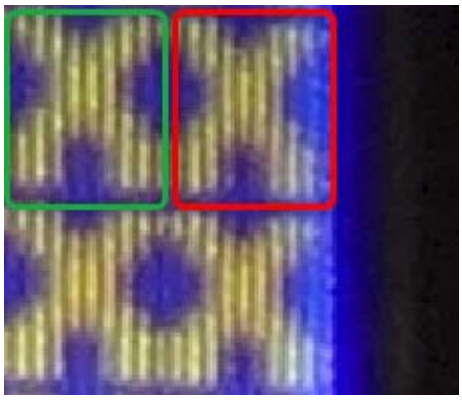
On peut constater des écarts de plusieurs pixels horizontaux et lignes verticales entre le centre de l'écran et les bords de ce dernier.

Verticalement, un écart de 5 lignes peut exister entre la première ligne visible dans un coin et la première ligne visible au milieu de l'axe horizontal de l'écran :

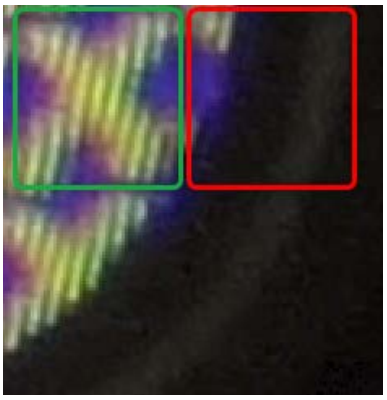


Horizontalement, un écart de presque 1 caractère CRTC peut être constaté entre une colonne située au niveau du coin latéral haut ou bas de l'écran et cette même colonne au milieu de l'axe vertical de l'écran (ligne 136) :

Colonnes au milieu à droite.



Mêmes colonnes dans le coin inférieur droit.



C'est en comptant le nombre de caractères horizontaux et de lignes verticales à partir du milieu des axes horizontaux et verticaux de l'écran qu'on peut déterminer les valeurs à programmer pour que l'intégralité des pixels visibles de l'écran soient « occupés ». Il faut toutefois garder à l'esprit que de nombreux pixels de ces lignes et colonnes ne seront pas visibles.

21.2 FULLSCREEN HORIZONTAL

Sur un moniteur CTM, on peut visualiser une ligne horizontale de **48 caractères CRTC**, ce qui représente 96 octets (192 pixels mode 0, 384 pixels mode 1, 768 pixels mode 2).

Le centrage d'une ligne horizontale dépend du signal HSYNC (voir chapitre 16, page 95).
Le premier caractère visible à gauche est le 15ème caractère à partir de la position C0=R2.

Pour obtenir un centrage exact sur une ligne de 64 µsec (R0=63), on doit positionner R2=50 (avec R3>=6) pour afficher C0=0 à l'extrême gauche de l'écran.

Selon le réglage H-HOLD du moniteur (accessible via un petit tournevis plat dans le petit trou à l'arrière gauche du moniteur), on peut voir apparaitre le BORDER à droite ou à gauche si la taille de la ligne est définie à 48 caractères.

Si R3 est inférieur à 6, alors l'écran est décalé à droite (voir chapitre 14.4, page 85).

21.3 FULLSCREEN VERTICAL

Sur un moniteur CTM, on peut visualiser **272 lignes** verticales à 50 Hz (et bien plus avec de l'interlace « maison » en abaissant la fréquence, mais c'est un autre sujet).

L'image commence à être visible à partir de la 34ème ligne, ce qui représente la deuxième ligne du 5ème caractère de 8 lignes à partir du début de la VSYNC. (voir chapitre 15, page 92).

Si R7 est positionné avec 35 alors que R4=38 (et R5=R8=0), alors 33 lignes « non visibles » seront générées à partir de la VSYNC, et la ligne C4=0/C9=1 sera visible en partie.

22 TRUCS ET ASTUCES

Dès qu'il est nécessaire de modifier les registres de plusieurs circuits, les 64 µsec d'une ligne sont souvent une contrainte importante à respecter. Plusieurs astuces permettent de gagner de précieuses µsec. Les principales optimisations résident dans l'économie de registres Z80A, la méthode d'accès aux entrées-sorties ou la gestion des itérations. Ces 3 approches sont combinables à loisir.

22.1 MISE A JOUR DE R12/R13

Lorsqu'il est question de modifier fréquemment les valeurs des registres de certains circuits,
Il n'est pas toujours nécessaire de modifier les 2 registres R12/R13 lorsque certaines ruses sont employées en temps critique, en général pour les démos.

Il est possible de n'affecter que l'un ou l'autre, ou même de manière dissociée selon l'architecture mémoire de ce qui doit être affiché.

Le mode d'accès au CRTC, qui implique la sélection préalable d'un registre, est très consommateur en CPU. (tout étant relatif...). Une option pour minimiser le temps d'accès à ces deux registres est d'utiliser l'instruction **OUT (nn),A** avec toutes les réserves qui entourent l'utilisation de cette instruction malicieuse sur un CPC.

Exemple 1 :
Le code suivant, par exemple, permet d'initialiser le couple R13/R12 rapidement (en tenant bien sûr compte du préchargement des registres). Le principe est de permettre la sélection d'un registre CRTC sans manipuler B plusieurs fois pour basculer sur la fonction d'écriture de données du circuit.
Prérequis : B=&BC, A=12, C=13, H=R12, L=R13
(facultativement, SP=&BABA)

```
OUT (C),C       ; Sélect R13
INC B           ; B=Crtc Data Address
OUT (C),L       ; R13=L
OUT (0),A       ; Select R12 (12="combo" CRTC Reg Select & Valeur Reg Select)
OUT (C),H       ; R12=H
```

Il est possible de sélectionner les registres CRTC 8, 4 et 0 avec cette méthode. Cependant les deux dernières valeurs envoient également la valeur du registre A sur le port A du PPI.

Exemple 2 : R0=0

```
LD B,#BD        ; B=Crtc Data Address
XOR A
OUT (0),A       ; Select R0
OUT (C),A       ; or OUT(C),0 (see chapter 22.4)
```

22.2 UTILISATION COMMUNE DE REGISTRE(S)

Dans le but d'économiser les précieux registres du Z80A, quelques astuces consistent à placer la valeur des ports d'entrée/sortie ou la valeur des registres de sélection dans des registres qui servent également de pointeurs sur des tables. Cela limite cependant l'accès à des tables dont le poids fort, par exemple, est une valeur valide pour le circuit.

Cela pose néanmoins une contrainte importante sur la carte mémoire.
Par exemple, il est possible de placer en mémoire aux adresses &BC00, &BD00, &BE00, &7F00 des tables contenant des valeurs utiles.

Exemple 1 :

```
LD B,#BD        ; BC est à la fois port d'écriture CRTC
LD A,(BC)       ; et pointeur sur la valeur à écrire (indexée par C ici)
OUT (C),A
```

Exemple 2:

```
LD B,#BE        ; BE est port d'écriture CRTC (via OUTI)
LD H,B          ; Mais également pointeur sur la valeur à écrire (indexée par L)
OUTI            ; HL étant également incrémenté dans l'opération
```

Ce sport peut également s'appliquer aux registres de sélection, comme 12 lorsqu'il est question de modifier R12 par exemple.

```
LD B,#BC
LD H,#0C        ; HL est un pointeur sur une table entre #0C00 et #0CFF
OUT (C),H       ; Mais H sert aussi de sélecteur de registre CRTC
INC B
INC B
OUTI            ; La valeur présente à l'adresse #C0xx est envoyée dans R12
```

22.3 ATTENTE DE VSYNC

Il est parfois intéressant que B soit déjà préchargé à l'issue d'une attente « traditionnelle » de VSYNC. Il est possible d'utiliser l'instruction **IN A,(0)** pour éviter que B soit impliqué, mais cela implique de recharger A avec l'adresse I/O du Port B (PPI) à chaque itération :

```
WSYNC   LD A,#F5                WSYNC   LD B,#F5
        IN A,(0)                        IN A,(C)
        RRA                             RRA
        JR NC,WSYNC                     JR NC,WSYNC
```

En chargeant A à l'aide d'un autre registre 8 bits préchargé, la boucle occupe alors le même temps qu'une boucle « standard » avec B car IN A,(0) s'exécute en 3 µsecondes.

22.4 VALEUR 0

Il existe une instruction non documentée (#ED,#71) intéressante qui permet d'envoyer la valeur 0 sur l'adresse d'I/O présente dans le registre B : **OUT (C),0**.
A priori, la valeur envoyée par cette instruction dépend du type de logique MOS du Z80A. Sur un NMOS, qui équipe les CPC, c'est 0 qui est envoyé. Sur un Z80 CMOS, c'est la valeur 255.
Exemple :

```
LD BC,#BC09     ; Voir chapitre 12.4.1, page 57
OUT (C),C
INC B
OUT (C),C
OUT (C),0
```

22.5 OUTI/OUTD ET REGISTRE D'ETAT

La documentation officielle est fausse concernant les bits N et C de du registre F du Z80A pour les instruction OUTI et OUTD. (c'est sans doute le cas aussi pour OTIR et OTDR, mais d'un intérêt limité sur CPC). Lorsque la somme de la valeur envoyée (présente en HL) au circuit et le registre L (post traitement (incrémenté/décrémenté)) est supérieur à 255, alors N=C=1. Cela permet potentiellement de tester la fin d'une table sans compteur mais en ayant une contrainte sur l'adresse de la table.

22.6 ITÉRATIONS ET BRANCHEMENT CONDITIONNELS

Lorsque plus aucune optimisation n'est possible, une méthode très employée est d'éliminer les tests et instructions de branchements de boucle. Cela peut cependant entrainer une hausse non négligeable de la ram nécessaire.

Au lieu de traiter des branchements conditionnels, une autre méthode classique consiste à utiliser une liste de pointeurs d'exécution via la pile. Chaque instruction **RET** (3 µsec) amenant le registre PC à basculer sur nouvelle routine (et SP à pointeur sur l'adresse suivante). Cette méthode nécessite en général de préparer la table en amont, notamment pour gérer les cas de sortie si cela n'est pas prévu par ailleurs. Cette méthode condamne néanmoins l'usage des interruptions (sauf à l'avoir prévu).

Une autre méthode consiste à utiliser les précieuses instructions **JP HL, JP IX ou JP IY**.
La première des 3 s'exécute en 1 µsec (les 2 autres avec 1 µsec de plus). L, LX ou LY peuvent servir d'index à condition de rester dans une page de 256 octets.
Remarque : La notation ZILOG JP (HL/IX/IY) est trompeuse, car le saut a lieu à l'adresse contenue dans le registre HL/IX/IY et non à l'adresse contenue à l'adresse pointée par HL/IX/IY. De nouveaux assembleurs (comme RASM, créé par Edouard Bergé (aka Roudoudou) ou ORGASM, créé par Yves Gerey (aka Madram)) acceptent ces nouvelles notations.

Enfin, et c'est très anecdotique, mais il est toujours possible d'utiliser les interruptions du CPC et s'arranger pour que R52 génère une interruption afin d'interrompre une boucle, en tenant compte qu'une instruction non répétitive n'est pas sécable.

22.7 PERDRE DU TEMPS…

Lorsqu'il est question d'attendre de manière précise, il est possible d'utiliser des séries de NOP (1 µsec) mais cela peut considérablement allonger le code présent en ram. Si l'objectif est de minimiser la taille d'une boucle en ram, il existe quelques instructions plus ou moins « neutres » pour les registres, qui occupent peu d'espace et permettent d'attendre plus longtemps que l'instruction NOP.

Attention aux interruptions et au contenu de la pile (ou registres) lorsque la neutralité des données est gérée avec des instructions associées à la pile.

Instruction (s)	Durée (en µsec)	Espace occupé (en octets)	Attention !
NOP	1	1	
CP (HL)	2	1	Modifie F
JR $+2	3	2	
INC HL + DEC HL	4	2	
INC (HL) + DEC (HL)	6	2	Modifie F
PUSH HL + POP HL	7	2	Modifie le contenu de la pile
EX (SP),HL + EX (SP),HL	12	2	

23 DUREE INSTRUCTIONS Z80A SUR CPC

Le Z80A est interrompu par le GATE ARRAY via sa broche READY lorsque ce dernier lit les 2 octets de la ram adressée par le CRTC à chaque µ-seconde. Le Z80A teste l'état de sa broche WAIT selon la définition des différents cycles de l'instruction lorsqu'il a besoin d'accéder à la ram ou effectuer une entrée-sortie. La broche WAIT (N°24) du Z80A est mise à l'état bas pendant 3 cycles sur 4, provoquant un alignement de chaque accès mémoire lorsque le signal WAIT est de nouveau haut. Le tableau suivant décrit la durée des instructions du Z80A en µsecondes.

Instruction	µs	Size		Instruction	µs	Size
ADC A,(HL)	2	1		DI	1	1
ADC A,(IX/IY+d)	5	3		EI	1	1
ADC A, A/B/C/D/E/H/L	1	1		EX (SP),HL	6	1
ADC A,HX/LX/HY/LY	2	2		EX (SP),IX/IY	7	2
ADC A,d	2	2		EX AF,AF'	1	1
ADC HL,BC/DE/HL/SP	4	2		EX DE,HL	1	1
ADD A,(HL)	2	1		EXX	1	1
ADD A,(IX/IY+d)	5	3		HALT	1	1
ADD A, A/B/C/D/E/H/L	1	1		IM m	2	2
ADD A, HX/LX/HY/LY	2	2		IN A/B/C/D/E/H/L,(C)	4	2
ADD A,d	2	2		IN A,(d)	3	2
ADD HL,BC/DE/HL/SP	3	1		IN F	4	2
ADD IX,BC/DE/HL/SP	4	2		INC (HL)	3	1
ADD IY,BC/DE/HL/SP	4	2		INC (IX/IY+d)	6	3
AND A,(HL)	2	1		INC A/B/C/D/E/H/L	1	1
AND A,(IX/IY+d)	5	3		INC HX/LX/HY/LY	2	2
AND A, A/B/C/D/E/H/L	1	1		INC BC/DE/HL/SP	2	1
AND A, HX/LX/HY/LY	2	2		INC IX/IY	3	2
AND A,d	2	2		IND	5	2
BIT x,(HL)	3	2		INDR	5/6	2
BIT x,(IX/IY+d)	6	4		INI	5/6	2
BIT x, A/B/C/D/E/H/L	2	2		INIR	4/5	2
CALL cond,aa	5/3	3		JP aa	3	3
CALL aa	3	3		JP cond,aa	3	3
CCF	1	1		JP HL	1	1
CP A, (HL)	2	1		JP IX/IY	2	2
CP A, (IX/IY+d)	5	3		JR a	2	2
CP A, A/B/C/D/E/H/L	1	1		JR cond,a	2/3	2
CP A, HX/LX/HY/LY	2	2		LD (BC/DE),A	2	1
CP A,d	2	2		LD (HL),A/B/C/D/E/H/L	2	1
CPD	5	2		LD (HL),d	3	2
CPDR	5/6	2		LD (IX/IY+d), A/B/C/D/E/H/L	5	3
CPIR	5/6	2		LD (IX/IY+d),d'	6	4
CPI	5	2		LD (aa),A	4	3
CPL	1	1		LD (aa),BC/DE/SP/IX/IY	6	4
DAA	1	1		LD (aa),HL	5	3
DEC (HL)	3	1		LD A,(BC/DE)	2	1
DEC (IX/IY+d)	6	3		LD A/B/C/D/E/H/L,(HL)	2	1
DEC A/B/C/D/E/H/L	1	1		LD A/B/C/D/E/H/L,(IX/IY+d)	5	3
DEC HX/LX/HY/LY	2	2		LD A,(aa)	4	3
DEC BC/DE/HL/SP	2	1		LD A/B/C/D/E/H/L, A/B/C/D/E/H/L	1	1
DEC IX/IY	3	2		LD HX/LX,B/C/D/E/HX/LX	3	3

Instruction	µs	Size	I/O	Instruction	µs	Size
LD HY/LY,B/C/D/E/HY/LY	3	3		RRA	1	1
LD BC/DE/HL/SP,dd	3	3		RRC (HL)	4	2
LD IX/IY,dd	4	4		RRC (IX/IY+d)	7	4
LD BC/DE/HL/SP/IX/IY,(aa)	6	4		RRC(IX/IY+d),A/B/C/D/E/H/L	7	4
LD HL,(aa)	5	3		RRC A/B/C/D/E/H/L	2	2
LD I,A	3	2		RRCA	1	1
LD R,A	3	2		RRD	5	2
LDD	5	2		RST 0/8/10h/18h/28h/30h/38h	4	1
LDDR	5/6	2		SBC A,d	2	2
LDI	5	2		SBC A,(HL)	2	1
LDIR	5/6	2		SBC A,(IX/IY+d)	5	3
NEG	2	2		SBC A, A/B/C/D/E/H/L	1	1
NOP	1	1		SBC A, HX/LX/HY/LY	2	2
OR A,(HL)	2	1		SBC HL,BC/DE/HL/SP	4	2
OR A,(IX/IY+d)	5	3		SCF	1	1
OR A, A/B/C/D/E/H/L	1	1		SET x,(HL)	4	2
OR A, HX/LX/HY/LY	2	2		SET x,(IX/IY+d)	7	4
OR A,d	2	2		SET x,(IX/IY+d),A/B/C/D/E/H/L	7	4
OTDR	6/5	2	5*	SET x,A/B/C/D/E/H/L	2	2
OTIR	6/5	2	5*	SLA (HL)	4	2
OUT (C),A/B/C/D/E/H/L	4	2	3	SLA (IX/IY+d)	7	4
OUT (C),0	4	2	3	SLA (IX/IY+d),A/B/C/D/E/H/L	7	4
OUT (d),A	3	2	3	SLA A/B/C/D/E/H/L	2	2
OUTD	5	2	5*	SLL (HL)	4	2
OUTI	5	2	5*	SLL (IX/IY+d)	7	4
POP AF/BC/DE/HL	3	1		SLL (IX/IY+d),A/B/C/D/E/H/L	7	4
POP IX/IY	4	2		SLL A/B/C/D/E/H/L	2	2
PUSH AF/BC/DE/HL	4	1		SRA (HL)	4	2
PUSH IX/IY	5	2		SRA (IX/IY+d)	7	4
RES x,(HL)	4	2		SRA A/B/C/D/E/H/L	2	2
RES x,(IX/IY+d)	7	4		SRL (HL)	4	2
RES x,(IX/IY+d),A/B/C/D/E/H/L	7	4		SRL (IX/IY+d)	7	4
RES x, A/B/C/D/E/H/L	2	2		SRL (IX/IY+d),A/B/C/D/E/H/L	7	4
RET	3	1		SRL A/B/C/D/E/H/L	2	2
RET cond	2/4	1		SUB A,(HL)	2	1
RETI	4	2		SUB A,(IX/IY+d)	5	3
RETN	4	2		SUB A,A/B/C/D/E/H/L	1	1
RL (HL)	4	2		SUB A, HX/LX/HY/LY	2	2
RL (IX/IY+d)	7	4		SUB A,d	2	2
RL (IX/IY+d),A/B/C/D/E/H/L	7	4		XOR A,(HL)	2	1
RL A/B/C/D/E/H/L	2	2		XOR A,(IX/IY+d)	5	3
RLA	1	1		XOR A,A/B/C/D/E/H/L	1	1
RLC (HL)	4	2		XOR A, HX/LX/HY/LY	2	2
RLC (IX/IY+d)	7	4		XOR A,d	2	2
RLC (IX/IY+d),A/B/C/D/E/H/L	7	4				
RLC A/B/C/D/E/H/L	2	2				
RLCA	1	1		x=[0..7] d=[0..ff]		
RLD	5	2		aa=[0..ffff] a=[0..ff]		
RR (HL)	4	2		m=[0..2]		
RR (IX/IY+d)	7	4		* some exceptions exist		
RR (IX/IY+d),A/B/C/D/E/H/L	7	4				
RR A/B/C/D/E/H/L	2	2				

24 CRTC 1 : STATUS REGISTER

24.1 GÉNÉRALITÉS

Ce registre de status, tel que décrit ci-dessous, est disponible **uniquement sur le CRTC 1**.
Sa lecture sur les CRTC 0 et 2 est déconseillée, surtout si c'est pour tester le type du CRTC.

L'adresse d'entrée/sortie de ce registre en lecture est &BE00.

CRTC	7	6	5	4	3	2	1	0
0	x	x	x	x	x	x	x	x
1	x	L	V	x	x	x	x	x
2	x	x	x	x	x	x	x	x
3	d	d	d	d	d	d	d	d
4	d	d	d	d	d	d	d	d

Haute impédance

Haute impédance

L	
1	Light pen en lecture
0	Le registre R16/R17 peut être lu

V	
1	BORDER R6 est vrai
0	BORDER R6 est faux

d : Miroir port BF00 (lecture reg CRTC)

Autres CRTC	7	6	5	4	3	2	1	0
UM6845E	U	L	V	x	x	x	x	x
R6545E	U	L	V	x	x	x	x	x

U	
1	Evènement de mise à jour
0	Le registre R31 a été lu/écrit par le MPU

24.2 FONCTION BORDER R6

Sur le CRTC 1, le bit 5 du registre de Status est mis à jour lorsque C0=R0 selon les conditions BORDER R6 (Faux: C4=C9=C0=0 / Vrai: C4=R6 & C9=C0=0).

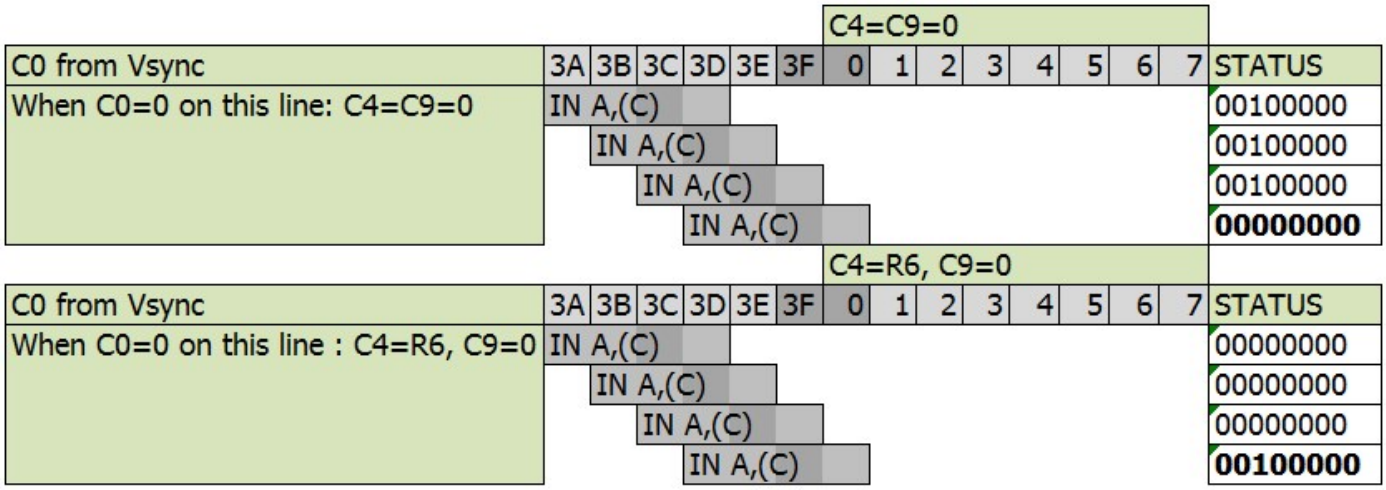
Le bit vaut 1 lorsque la condition BORDER R6 est vraie.
Le bit vaut 0 lorsque la condition BORDER R6 est fausse.

Cela ne signifie par forcément que du BORDER ou des CARACTERES sont affichés, car cette évaluation est faite lorsque C0=R0.

Or du BORDER peut déjà être affiché si la condition BORDER R1 est vraie.

A noter également que si R6 est positionné à 0 (alors que C4>0) pour générer du BORDER, cet état n'est pas détecté. La valeur 0, comme d'autres registres du CRTC 1, est gérée de manière particulière. Autrement dit, si R6=0 alors que C4>0 et que le statut valait 0 (Caractères affichés), le bit 5 du registre de status continuera de valoir 0.

Les schémas sur la page suivante décrivent le passage exact à 1 ou 0 de ce bit de status.



24.3 DUMMY REGISTER

Parmi quelques uns des mythes et légendes à propos des CRTC du CPC, figure l'existence du registre 31, appelé DUMMY REGISTER.

Si ce registre 31 existe bien sur le CRTC UM6845E de la société UMC, **il n'existe pas** sur les CRTC UM6845R (type 1), ni sur les CRTC UM6845 (type 0).

Sur le CRTC UM6845R, le bit 7 du registre de status qui y fait référence est simplement inutilisé.
Sur le CRTC UM6845, le registre de status lui-même n'existe pas (de là à trouver son bit 7...).

Sur le CRTC UM6845E (hors CPC), le registre 31 est en relation avec le mode transparent.

La gestion de ce mode utilise également les bits 6 et 7 du registre 8 en plus du bit 7 du registre de status. Je ne m'étendrais pas sur ce sujet, qui est aussi intéressant que la programmation du EF9345P...

25 INTERRUPTIONS

25.1 GENERALITES

Les interruptions sont générées sur CPC par le GATE ARRAY, selon des paramètres définis par le CRTC.

Le GATE ARRAY dispose d'un compteur dont l'objectif est de compter de 0 à 51 avant de repasser à 0, qui est souvent appelé **R52**.

Ce compteur est incrémenté à la fin d'une HSYNC qui est produite selon les paramètres définis par les registres R0, R2 et R3 du CRTC.

En fonctionnement "standard", avec une HSYNC programmée toutes les 64 µsec, 1 interruption peut avoir lieu toutes les 3328 µsec (300 Hz).

Sur un CPC au standard Européen, cela représente 6 interruptions par frame :
6 x 52 lignes = 312 lignes.

25.2 GESTION DU COMPTEUR R52

Il existe plusieurs particularités de gestion de la valeur du compteur **R52** par le GATE ARRAY.

Il peut repasser à 0 dans les circonstances suivantes :

- Lorsqu'il dépasse 51.
- En positionnant à 1 le bit 4 du registre MF du GATE ARRAY.
- A la 2ème HSYNC après le début de la VSYNC.
- Lorsqu'une interruption était en attente et que le bit 5 de **R52** est égal à 1.

Son contenu peut être "corrompu" (le bit 5 du compteur repasse à 0) :

- Lorsqu'une interruption était armée (en attente) et qu'elle est autorisée alors que **R52** avait continué à évoluer.

25.3 CONDITIONS DE DÉCLENCHEMENT

Pour qu'une interruption puisse survenir, il faut qu'elle soit armée (un flag de requête d'interruption est positionné).

Elle est armée lorsque R52 repasse à 0.

- Si les interruptions sont autorisées, alors une interruption a lieu.
- Si les interruptions ne sont pas autorisées, le compteur continue à s'incrémenter, et l'interruption reste armée.
  Lorsque l'interruption est autorisée (via l'instruction EI du Z80A) alors qu'elle est armée, alors :
  - Une interruption se produit après l'instruction qui suit le EI. [Un HALT suivant un EI dans cette circonstance sera considéré comme 1 NOP]
  - Le bit 5 du compteur R52 est remis à 0. Dans cette circonstance, si le bit 5 du compteur avait atteint 1 (R52>=32), alors cela revient à "retirer" 32 lignes du compteur courant.

Ce qui a pour effet de décaler le moment ou la prochaine interruption pourra se produire.

Elle est armée également lors de la deuxième HSYNC après le début de la VSYNC, lorsque R52 est remis à 0, mais seulement si le bit 5 de R52 valait 1 (R52>=32). Ce mécanisme rudimentaire empêche que deux interruptions puissent se produire dans un intervalle trop rapproché.

Remarque 1 :
L'instruction Z80A EI sert à autoriser une interruption à se produire dès que cette dernière est armée. Lorsqu'une interruption se produit, les autres interruptions sont désactivées (même effet qu'un DI) jusqu'à ce qu'un nouvel EI survienne. Le compteur R52 continue cependant d'évoluer et si il repasse à 0, une nouvelle interruption est alors en attente, et va se produire dès que les interruptions seront autorisées de nouveau. Une seule interruption peut être en attente.
Afin d'éviter des problèmes de réentrance dans un code d'interruption, les concepteurs du Z80A ont imposé qu'une nouvelle interruption ne puisse pas survenir sur l'instruction suivant le EI.
Ceci a plus précisément été fait pour permettre aux instructions RET, RETI, RETN de s'exécuter juste derrière l'instruction EI, afin d'éviter que la pile déborde.

Remarque 2 :
Une séquence répétitive de l'instruction EI ne permettra pas la survenue d'une interruption avant la fin de cette séquence, chaque EI différant l'interruption.

Remarque 3 :
L'instruction Z80A HALT permet d'attendre qu'une interruption survienne en répétant indéfiniment des NOP de 1 µsec. Si les interruptions ne sont pas autorisées lorsque cette instruction s'exécute, le Z80A reste bloqué sur cette instruction. Cette instruction est intéressante dans la mesure où elle permet de caler une interruption à la microseconde près. En effet, une interruption ne peut pas couper une instruction (sauf une instruction répétitive comme LDIR ou OTIR).
Si l'instruction dure plusieurs microsecondes, l'interruption doit attendre la fin de l'instruction pour se produire. A ma connaissance, peu de jeux ont utilisé cette instruction. On peut cependant noter que le jeu Trailblazer (edité par Gremlin Graphics en 1986) cale ses « splitraster » avec un HALT.

25.4 INTERRUPTION MODE 1

La plupart des codes écrits pour le CPC AMSTRAD "OLD" utilisent le mode IM 1 du Z80A.

Dans ce mode, lorsqu'une interruption survient, le code est interrompu avec une instruction équivalente à RST #38.

L'adresse suivant l'instruction interrompue (ou l'adresse de l'instruction si il s'agit d'une instruction répétitive) est mise sur la pile et PC=#38.

L'instruction Z80A RST #38 dure 4 µsec lorsqu'elle est appelée par du code.

Lorsqu'une interruption se produit, l'appel en #38 dure 5 µsec.

[pour le tester, il suffit de comparer le temps pris par un RST #38 et le temps pris par une interruption avec un code en temps fixe sur 19968 nop]

25.5 INTERRUPTION MODE 2

Le mode IM 2, aussi appelé mode vectorisé, est prévu pour permettre à plusieurs périphériques de générer des interruptions, via une table de pointeurs sur des routines d'interruption.

A ma connaissance, aucun jeu n'a jamais utilisé ce mode d'interruption, sauf peut-être ceux qui sont sortis à partir des années 90.

L'adresse de la table est définie, pour le poids fort de l'adresse, par le registre I du Z80A.
Le poids faible de l'adresse est normalement défini par un des 128 périphériques possibles en indiquant son numéro sur les 7 bits de poids fort, le bit 0 valant 0.

Sur CPC AMSTRAD "OLD", la valeur de poids faible de l'adresse est indéterminée (valeur de Haute Impédance) et peut varier d'un CPC à l'autre.

[A vérifier: Est-ce que cette valeur peut varier tant que le CPC est sous tension, mais l'intérêt en gain de ram est modéré et conditionné]

Dans la perspective de déplacer l'adresse d'interruption, il est néanmoins possible d'utiliser ce mode moyennant quelques précautions :

Il faut pour cela créer une table de vecteurs d'interruption contenant 257 octets de même valeur.
En effet, le bit 0 de l'adresse de la table sélectionné étant imprévisible, cela peut amener le Z80A à aller lire son vecteur sur le 256ème et 257ème octet de la table.

Dans le cas ou le bit 0 vaudrait 0, le poids fort et faible du vecteur lu dans la table serait inversé par rapport à un vecteur lu sur une table ou le bit 0 vaut 1.

Il est donc conseillé de créer un vecteur d'interruption ou le poids fort du vecteur est égal à son poids faible.

Exemple : Table de vecteurs créée en #2000, qui contient entre #2000 et #2100, 257 fois la valeur #CA. L'interruption se produisant alors en #CACA (prout!)

Lorsqu'une interruption se produit, l'appel du pointeur situé dans la table dure 7 µsec.

25.6 CRTC & INTERRUPTIONS...

25.6.1 GÉNÉRALITÉS

Lorsque la condition interne C0=R2 est remplie, une HSYNC débute.

Sur les CRTC 0, 1, 2, la HSYNC débute visuellement 1 µsec environ avant l'affichage du caractère CRTC correspondant. ["environ" car visuellement, l'absence d'affichage de la HSYNC n'est pas exactement localisé en frontière de mot].

L'affichage des caractères par le GATE ARRAY sur ces CRTC recommence à partir de C0 Disp=R2+R3-1 (sauf pour R3=0 pour les CRTC 0 et 1).

La mesure du moment où débute l'interruption est réalisée en considérant C0vs à partir du signal VSYNC renvoyé par le PPI.

Sur les CRTC 3 et 4, la HSYNC débute très exactement au début de l'affichage par le GATE ARRAY du caractère CRTC correspondant à C0=R2.

En règle générale, une interruption débute toujours 1 µsec après la fin de la HSYNC quelque soit le CRTC.

Etant donné que la HSYNC débute 1 µsec plus tôt que prévu sur un CPC sans ASIC (CRTC 0, 1, 2) il existe donc un décalage de 1 µsec avec une interruption sur un CPC avec ASIC (CRTC 3, 4).

Avec une même programmation de R2 et R3, une interruption se produit 1 µsec plus tard sur un CRTC 3 ou 4 que sur les autres CRTC, car l'ASIC gère une HSYNC synchrone avec l'affichage.

Je l'ai déjà écrit, mais la HSYNC générée par un ASIC est plus conforme au "retard" de gestion d'affichage opéré par les GATE ARRAY/ASIC.

Les schémas suivants décrivent cette gestion selon les CRTC.

25.6.2 CRTC 0, 1, 2

R3=14					R2																							
C0 from Vs	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
C0 from GA	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
				C3:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Code interrupted 15 µsec after C0vs=R2										

R3=8					R2																							
C0 from Vs	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
C0 from GA	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
				C3:	1	2	3	4	5	6	7	8	Code interrupted 9 µsec after C0vs=R2																

R3=1					R2																							
C0 from Vs	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
C0 from GA	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
				C3:	1	Code interrupted 2 µsec after C0vs=R2																							

25.6.3 CRTC 0, 1

R3=0					R2																							
C0 from Vs	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
C0 from GA	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
					No interruption																							

25.6.4 CRTC 2

R3=0					R2																								
C0 from Vs	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
C0 from GA	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
				C3:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Code interrupted 17 µsec after C0vs=R2								

25.6.5 CRTC 3, 4

R3=14					R2																								
C0 from Vs	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
C0 from GA	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
					C3:	1	2	3	4	5	6	7	8	9	10	11	12	13	14		Code interrupted 16 µsec after C0vs=R2								

R3=8					R2																								
C0 from Vs	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
C0 from GA	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
					C3:	1	2	3	4	5	6	7	8		Code interrupted 10 µsec after C0vs=R2														

R3=1					R2																								
C0 from Vs	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
C0 from GA	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
					C3:	1		Code interrupted 3 µsec after C0vs=R2																					

R3=0					R2																								
C0 from Vs	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
C0 from GA	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
					C3:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16		Code interrupted 18 µsec after C0vs=R2						

Remarque :

Dans la mesure où il est possible de générer des HBL "non actives" pour le moniteur (lorsque R3<=2), cela implique la possibilité de générer des interruptions dans la zone visible.

25.6.6 MISE EN PERSPECTIVE

Le schéma ci-dessous permet une mise en perspective d'une interruption programmée dans les mêmes conditions selon les différents CRTC.

Il permet d'appréhender les mécanismes en œuvre à partir du caractère **C0vs** de référence, compte tenu des différents délais mis en œuvre.

Lorsqu'il est question de compatibilité entre les CRTC, il faut donc vérifier les différences relatives aux délais de prises en comptes des registres et l'effet de certaines valeurs, notamment si ces registres sont modifiés lorsqu'une HSYNC est en cours.

	R3=14					R2																								
	C0 from Vs	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
CRTC 0, 1,	C0 from GA	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
						C3:	1	2	3	4	5	6	7	8	9	10	11	12	13	14		Code interrupted 15 µsec after C0vs=R2								
CRTC 3, 4	C0 from GA	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
						C3:	1	2	3	4	5	6	7	8	9	10	11	12	13	14		Code interrupted 16 µsec after C0vs=R2								
							1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16									

26 IDENTIFICATION CRTC

Il existe une pléthore de méthodes permettant d'identifier un CRTC, la plupart avec du code, d'autres uniquement visuellement.

Certaines méthodes impliquent parfois une désynchronisation de l'écran et sont donc pour cette raison moins bien considérées par certains puristes.

Il est aussi possible d'identifier un CRTC en étant capable d'identifier la machine, comme c'est le cas avec le CPC+, qui présente des différences de gestion sur d'autres circuits (et possède, le cas échéant, une page de fonctions étendues).

26.1.1 VIA LE DEBORDEMENT DE C4 ET/OU C9

Il est possible de faire déborder C4 dans de nombreuses situations différentes selon les CRTC, que ce soit en programmant R0, R9, R4, R7 ou même R8.
En général, sur CRTC 0, par exemple, C4 est incrémenté par défaut.
En gestion additionnelle, la valeur de C4 dépasse la valeur programmée dans R4.
Sur un CRTC 0, ce dépassement aura lieu 1 fois.
Sur un CRTC 1 ou 2, ce dépassement aura lieu plusieurs fois selon le contenu de R9 et R5.
Sur un CRTC 3 ou 4, il n'y aura pas de dépassement.
Si C4 atteint R7, alors une VSYNC peut se produire.

Exemple : Ecran de 312 lignes tel que R4=36, R9=7, R5=16
312=((36+1)x(7+1))+16
Sur un CRTC 3 ou 4, si R7>36, la VSYNC ne se produit plus.
Sur un CRTC 0, si R7>37, la VSYNC ne se produit plus.
Sur un CRTC 1 ou 2, si R7>39, la VSYNC ne se produit plus.

26.1.2 VIA LA GESTION VSYNC DURANT LA HSYNC

Sur un CRTC 2, positionner R2 et R3 de manière à ce que la VSYNC survienne durant la période de la HSYNC « désactive » la VSYNC pour le reste de la période C4=R7.
Il suffit donc de placer R2+R3 de manière à ce qu'il dépasse la valeur de R0.

26.1.3 VIA LA PRISE EN COMPTE DE LA VSYNC

Lorsque la VSYNC est déclenchée en positionnant R7 avec C4, lorsque C0>0, le compteur de ligne de la VSYNC est différent selon les CRTC. Il part de 0 sur un CRTC 0, et de 1 sur un CRTC 1, alors qu'aucune VSYNC ne survient dans cette condition sur un CRTC 3 ou 4.

26.1.4 VIA LA LONGUEUR DE LA VSYNC

Seul les CRTC 0, 3 et 4 permettent de gérer la longueur de la VSYNC.
Une VSYNC sur CRTC 1 et 2, qui démarre lorsque R7 était programmé avant que C4=R7, dure 16 lignes.

26.1.5 VIA LA LONGUEUR DE LA HSYNC

Sur les CRTC 2, 3 et 4, la HSYNC dure 16 µsec lorsque R3=0.
Sur les CRTC 1 et 2, il n'y a pas de HSYNC lorsque R3=0.
Il n'y a donc pas d'interruption, ce qui est une des méthodes permettant de tester les conséquences de la valeur 0 sur R3.

26.1.6 VIA DU BORDER, VISUELLEMENT

a) Un CRTC 2 dont C0=0 pendant la HSYNC ne peut plus désactiver son BORDER.
b) Un CRTC 0 et 2, dont R6=0 est en conflit lorsque C4=C9=0, alterne des octets de fond et de BORDER (à minima sur la première ligne sans programmation adéquate).
c) Un CRTC 0 et 2 va créer un octets de BORDER lorsque C0 atteint R0 (et R1>R0).
d) Les CRTC 0, 4 et 4 peuvent désactiver le BORDER (ou ajouter des délais sur la prise en compte du BORDER) avec la fonction SKEW BIT de R8, alolrs que cette fonction n'existe pas sur les CRTC 1 et 2.

26.1.7 VIA LE MODE INTERLACE

Les méthodes de comptage de C4 sont différentes selon les CRTC.
Sur CRTC 2, le comptage de C4 n'est pas affecté. Etant le seul, il est donc détectable une fois les autres identifiés.

26.1.8 VIA LE REGISTRE DE STATUS &BE00

Uniquement sur le CRTC 1, il est possible de tester le passage du bit 6 au moment opportun.

26.1.9 VIA LE REGISTRE DE LECTURE &BF00

Sur CRTC 0, il est possible de lire R12 et R13, ainsi que R14/R15 (curseur) et R15/R17 (stylo optique).
Sur CRTC 1 et 2, ce registre est inopérant et semble toujours renvoyer 0 (à confirmer).
Sur CRTC 3 et 4, il est possible de lire les registre R12, R13, R16 et R17. Le numéro de registre lu est cependant tronqué sur 3 bits. Lire R4 ou R20 revient donc à lire R12.
Sur CRTC 3, la lecture de R10 et R11 (ou équivalence pour les 3 premiers bits) renvoient des valeurs internes à l'ASIC (voir chapitre dédié).

26.1.10 VIA LES OVERSCAN BITS

Ces bits n'existent pas sur les CRTC du CPC de Madram, qui fonctionnent comme des PPI.

27 IDENTIFICATION CPC

Il est possible d'identifier des modèles de CPC à partir de différences qui ne sont pas liées au CRTC ou au contenu de la ROM.
C'est le cas pour les 2 machines qui disposent d'ASIC :

CPC PLUS	ASIC 40489	CRTC 3
CPC LOWCOST	ASIC 40226	CRTC 4

27.1 MÉTHODES D'IDENTIFICATION

27.1.1 ACTIVATION DES FONCTIONS ÉTENDUES

CPC PLUS : Activation via une séquence de délockage envoyée au CRTC
CPC LOWCOST : Pas de fonctions étendues (sans la séquence secrète).
AUTRES CPC : Pas de fonctions étendues.

27.1.2 BUG PPI PORT C

CPC PLUS : Lorsque le registre de commande est utilisé pour configurer le port C, les bits de ce port sont remis à 0. L'ASIC émule mal le PPI et ne remet pas ces bits à 0. Très génant pour la compatibilité des routines clavier entre le CPC PLUS et anciens modèles.
CPC LOWCOST: Le PPI 8255 n'est pas émulé par l'ASIC 40226. Ces CPC disposent d'un PPI et se comportent donc comme les CPC de première génération.
AUTRES CPC : Les bits du port C sont remis normalement à 0 par le registre de commande.

27.1.3 BUG PPI PORT B

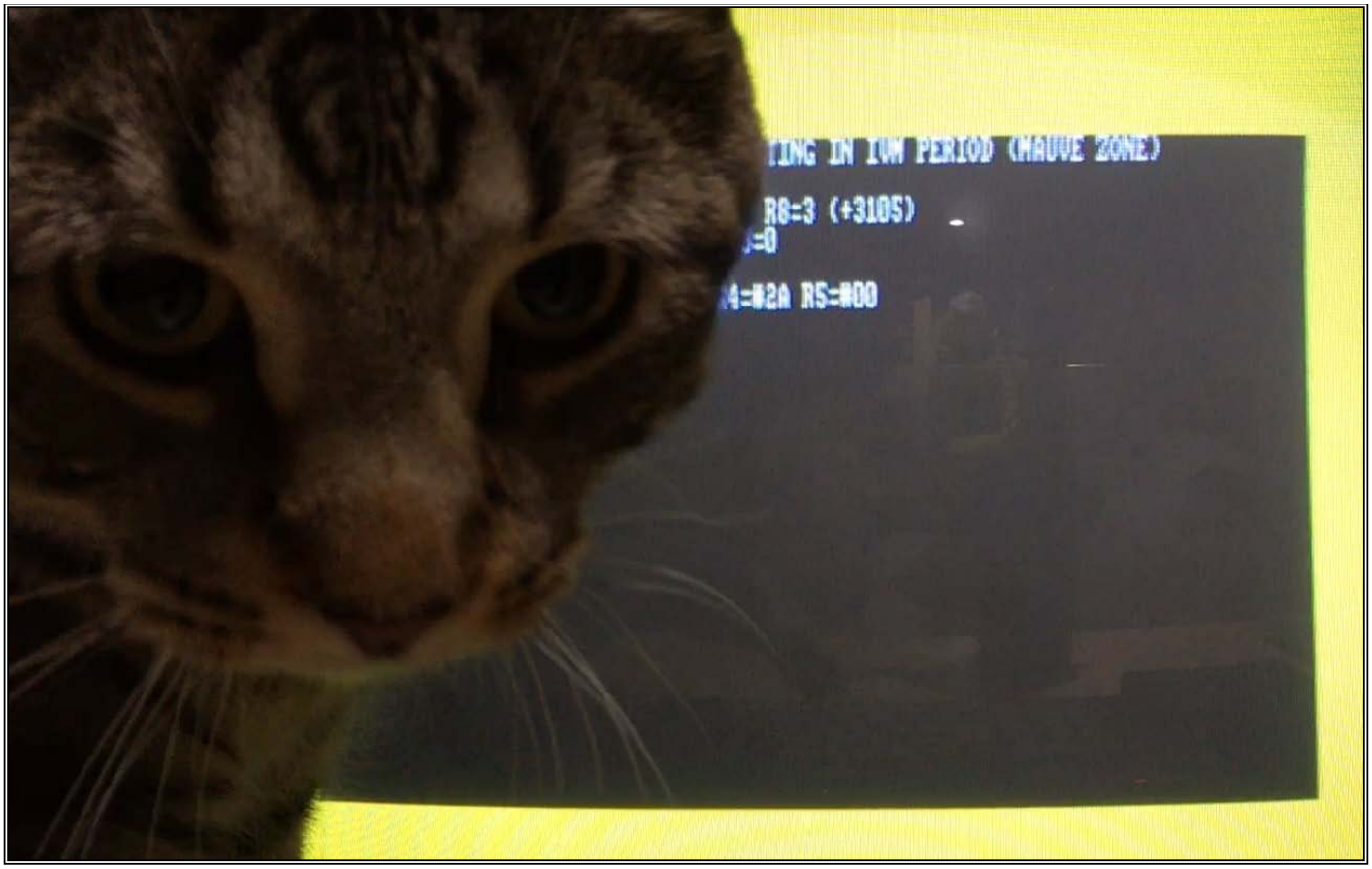
Sur un PPI, on peut programmer le sens du port B (entrée ou sortie)
En sortie, on peut donc y loger une valeur, qu'on pourra relire.

Le PPI du CPC LOWCOST ne permettrait pas de relire une valeur stockée dans le port B placé en sortie. On relit donc systématiquement la valeur du port B en entrée (et donc les périphériques qui y sont reliés).
Ceci reste à vérifier cependant, à défaut d'indiquer quels PPI étaient installés dans les CPC de test.

AMSTRAD a utilisé, tout comme pour les CRTC, des PPI 8255 de plusieurs fabricants.
On peut dénombrer les circuits suivants, utilisés dans tous les modèles indistinctement (464, 664, 6128)

- NEC D8255AC-2
- NEC D8255AC-5
- TOSHIBA TMP8255AP-5

A noter que la différence entre « -2 » et « -5 » tient à la fréquence maximale gérable par le circuit (4 Mhz pour le « -5 » et 5 Mhz pour le « -2 »)



Raaahhhhhhhhh !!
Con de Chat Canadien Curieux devant un CPC