Manuel de Référence Hardware
pour la
Console SEGA Mark III

Table des matières

La Console de Jeux Sega Mark III

Ce manuel décrit le hardware de la console de jeux SEGA Mark III. Le manuel est divisé en cinq sections:

• Le CPU.
• Le Processeur d'affichage video (VDP:Video Display Processor).
• Le système de gestion mémoire.
• Le système d'entrées/sorties.
• Le générateur de sons programmable (PSG:Programmable Sound Generator)

Chaque section commence par une description générale, suivie par la description détaillée de chaque bit de contrôle.

RUBRIQUES

Les sujets importants à l'intérieur de chaque section sont marqués comme le titre ci-dessus avec un "en-tête". Cela permet de retrouver rapidement un sujet spécifique.

REFERENCES ANTICIPEES

Dans certaines sections il est impossible de décrire le système sans faire de référence à des détails techniques décrits plus tard dans le manuel. Par exemple vous trouverez une référence aux "Interruption de Sprites" dans la section CPU, avant que la notion de sprite soit expliquée dans la section VDP.

Pour cette raison vous pourrez trouver utile de parcourir la première partie de chaque section pour vous faire une idée globale du système avant de vous plonger dans les descriptions détaillées.

On fera par la suite référence au système sous le terme "Mk3".

LE CPU

La Mk3 utilise un microprocesseur Z80A cadencé à 3,58 MHz.

L'implémentation des caractéristiques du Z80 ayant trait à la Mk3 est décrite dans cette section.

RESET

Le Z80A exécute un cycle de RESET quand l'appareil est allumé. C'est le seul mécanisme de reset implémenté.

Le bouton RESET sur la console ne contrôle pas le reset du Z80A, il est relié à un port d'entrée pour être testé par logiciel.

SYSTEME D'INTERRUPTION

La Mk3 implémente deux des interruptions du Z80: l'interruption non masquable (NMI:*Non-maskable Interrupt*) et l'interruption masquable (INT).

NMI

La broche NMI est connectée au bouton "PAUSE" de la console. L'appui sur ce bouton provoque l'exécution d'une instruction de redémarrage (*Restart*) à l'adresse $66. Cela agit comme un simple appel de sous-routine : le Pointeur de Code (PC) est déposé sur la pile et un saut à l'adresse $0066 est effectué.

Cette interruption est déclenchée sur front, ce qui signifie que si le bouton PAUSE est maintenu appuyé, vous n'aurez qu'une seule interruption.

NOTE: La dernière instruction de votre routine NMI doit être "RETN", Retour d'interruption non masquable.

Le bouton PAUSE est réservé à l'usage exclusif de votre programme; vous devez passer le contrôle à votre routine de pause en $0066.

La routine de pause la plus commune va basculer un *flag* (booléen) de pause et couper le son quand ce flag sera allumé. La routine INT, qui est activée périodiquement, vérifie alors l'état du flag de pause et ne fera aucun traitement si celui-ci est à 1 (elle boucle sur le flag de pause).

Cette méthode permet d'implémenter la fonction pause sans activer et désactiver les interruptions (avec une instruction DI-*Disable Interrupt*- et ultérieurement EI-*Enable Interrupt*). Cette combinaison DI-EI peut provoquer une interruption parasite dans le Z80.

La NMI ne peut être désactivée.

INT

Cette interruption est activée avec une instruction "EI" et désactivée avec une instruction "DI".

Le hardware de la Mk3 supporte seulement le "mode d'interruption 1". La *boot ROM* (ROM de démarrage) exécute l'instruction "IM 1" lors de la mise sous tension.

Dans le mode 1, l'activation de la broche INT du Z80A provoque l'exécution d'une instruction de redémarrage à l'adresse $38 à condition que l'interruption soit activée (*enabled*).

Cette interruption peut être activée par deux événements dans le circuit VDP:

1. Intervalle de blanking vertical (*Vertical Blanking Interval*).
2. Compteur de ligne horizontal (*Horizontal Line Counter*).

Voici le squelette d'une routine de gestion d'interruption:

```
        (en $0038)

          JP      INT
          .
          .
INT:
          PUSH    AF
          IN      A,(0BFH)
          .
          .
          .
          POP     AF
          EI
          RET
```

La première étape est de sauvegarder les registres et les flags de travail. D'autres instructions "PUSH" devront être ajoutées pour chaque registre que votre interruption utilisera.

La lecture du port d'entrée/sortie $BF engendre deux actions. Primo, cela efface la ligne de requête d'interruption provenant du circuit VDP. Secundo, cela fournit les informations suivantes à propos du VDP:

```
bit 7 --- 1: Interruption VBLANK, 0:Interruption H-Line (si activée).
bit 6 --- 1: 9 sprites sur une ligne de raster.
bit 5 --- 1: Collision de sprites.
Bits 4-0  (sans signification)
```

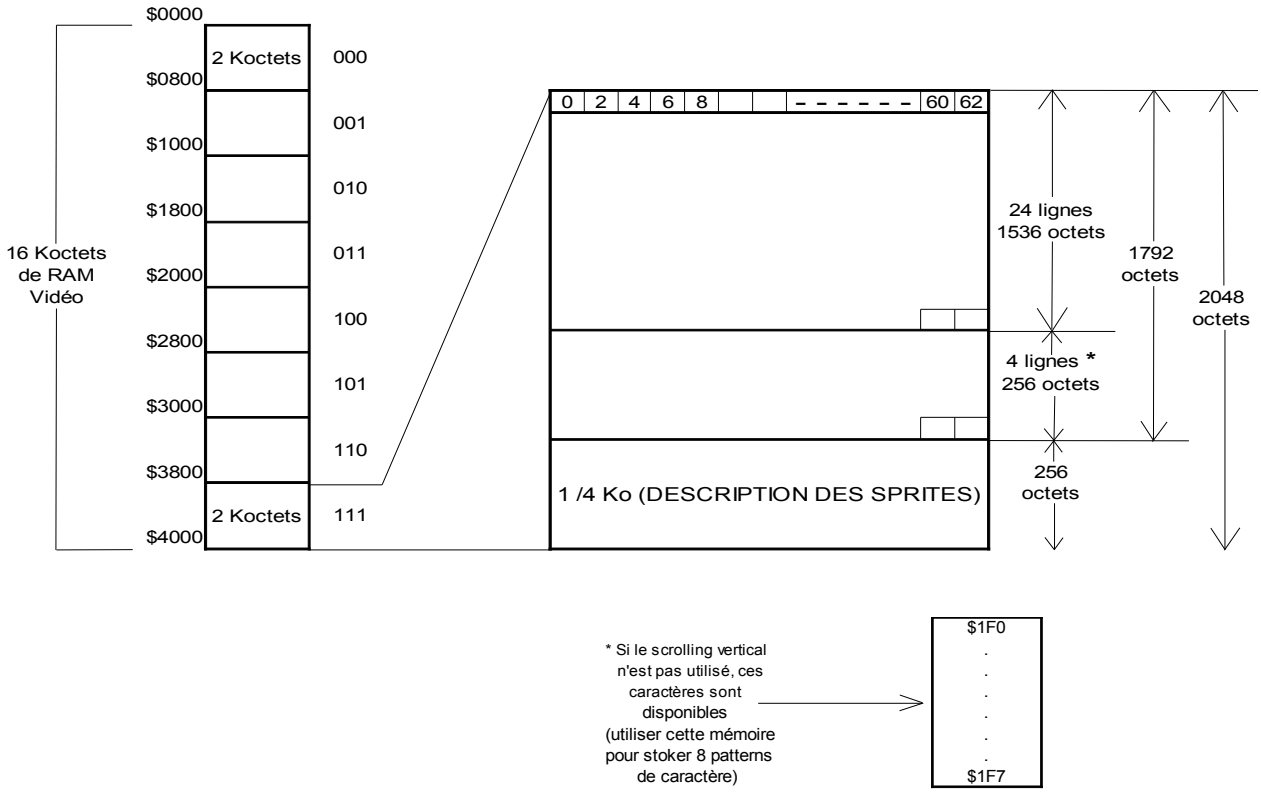
Les interruptions du VDP VBLANK et H-Line (lignes de raster) sont activées (*enabled*) par les bits IE et IE1 dans les registres VDP 0 et 1, suivant le schéma suivant:

Interrupt Enable Bits

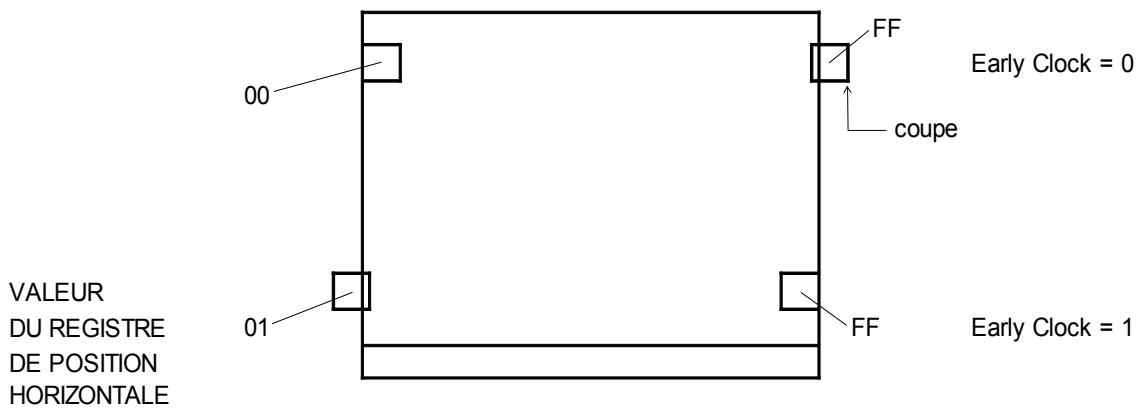
IE (R1, bit 5)	IE1 (R0, bit 4)	Source d'interruption
0	1	Interruption H-Line seulement.
1	1	A la fois H-Line et VBLANK.

Quand une INT est acceptée, le système d'interruption est désactivé, exactement comme si une instruction "DI" avait été exécutée. Le code pour sortir d'une routine d'interruption doit donc restaurer les registres, exécuter une instruction "EI", et faire un return.

Map d'écran



Early Clock Bit
(bit d'Horloge anticipé)



LE PROCESSEUR D'AFFICHAGE VIDEO (VDP)

La Mk3 utilise un circuit contrôleur de vidéo perfectionné appelé VDP (*Video Display Processor*). Le VDP est un circuit conçu sur mesures par Sega dont l'architecture est similaire à un TMS9918A amélioré.

La résolution d'écran du VDP est de 256 pixels horizontaux et de 192 pixels verticaux. Un pixel peut être affiché de 16 couleurs différentes sélectionnées dans une palette de 64.

16 Kilooctets de RAM sont dédiés au système vidéo. Cette RAM (appelée RAM vidéo) est reliée directement au circuit VDP et n'apparaît pas dans l'espace mémoire du Z80. Le Z80 lit et écrit dans la RAM vidéo par l'intermédiaire des registres du VDP.

Le VDP intègre deux systèmes graphiques indépendants: un système de *background* (arrière-plan) et un système de sprites.

ORGANISATION DE LA RAM VIDEO

La RAM Vidéo de 16 Kilooctets est divisée en 3 sections:

1. Une *map* (carte) de 1792 octets. Cette map détermine l'emplacement de caractères (aussi appelées *tiles*) sur la grille de background de 32x24 éléments.

2. Une table d'attributs des sprites de 256 octets. Cette table fixe les coordonnées X-Y et le numéro de caractère de 1 à 64 objets déplaçables ou sprites.

3. Un générateur de caractères de 14336 octets. Ce sont des *patterns* (motifs) de caractères de 8x8 pixels pour le background, et/ou des patterns de 8x8 ou 8x16 pixels pour les sprites.

32 octets définissent un seul caractère de 8x8 pixels. La portion "générateur de caractères" de la RAM Vidéo permet de définir jusqu'à 448 caractères.

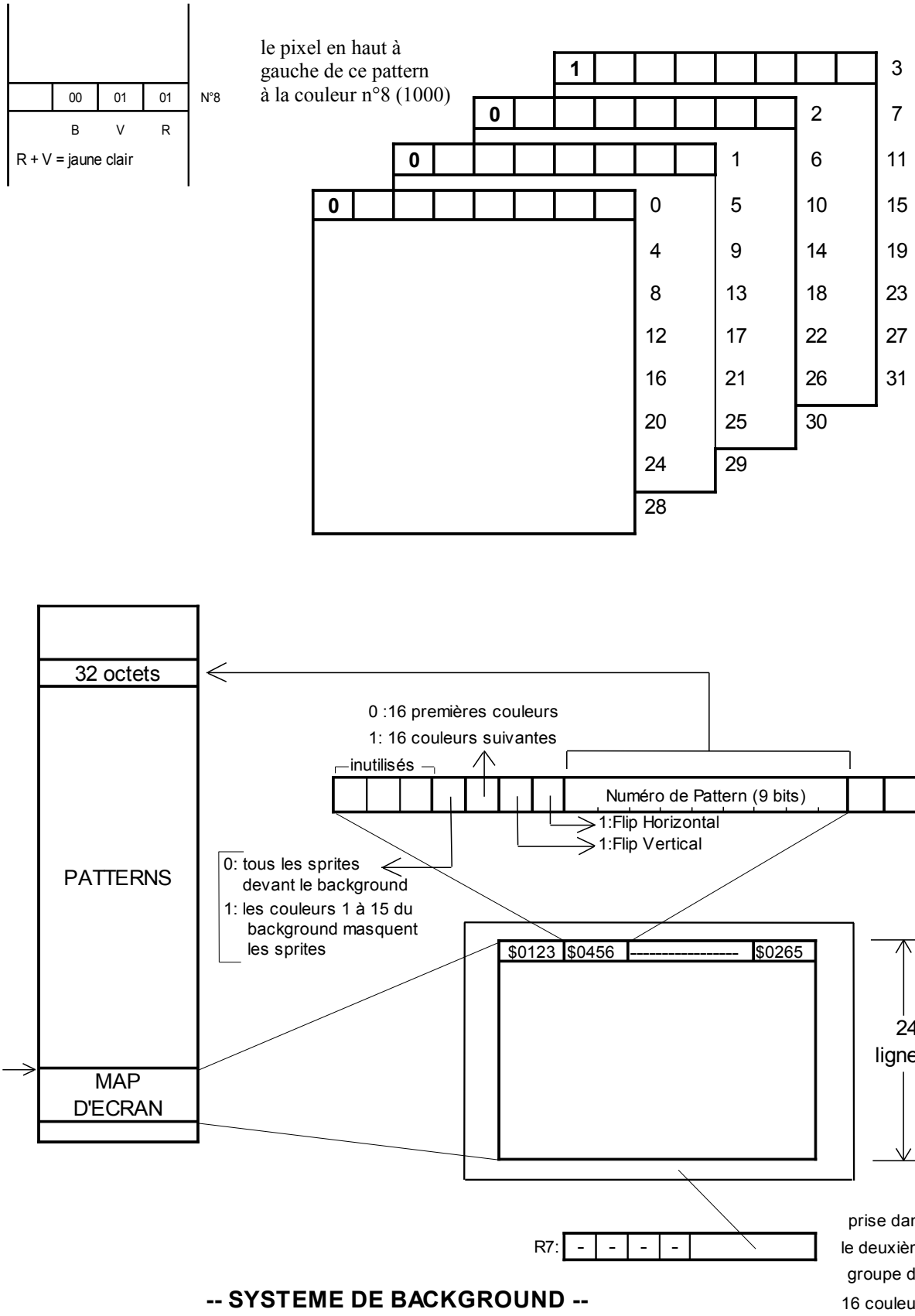
L'emplacement des 3 sections de la RAM Vidéo est contrôlé par des registres du VDP. Comme nous le verrons, vous pouvez choisir où elles apparaissent dans la map de la RAM Vidéo.

COULEUR

L'attribut de couleur est porté par les patterns de caractères dans la portion "générateur de caractères" de la RAM Vidéo. Chaque pixel dans le pattern de caractère contient 4 bits d'information. Un choix parmi 16 couleurs est donc possible pour chaque pixel.

Les 16 couleurs disponibles ne sont pas fixes, elles sont stockées dans une *Color RAM* (RAM de Couleurs) à l'intérieur du VDP. La Color RAM est organisée en 32 valeurs de 6 bits. Les six bits fournissent 4 niveaux (2 bits) de rouge, de vert et de bleu.

Notez que 32 valeurs de couleur représentent deux fois l'espace adressable par des pixels de 4 bits. D'autres bits de contrôle du VDP permettent la sélection du groupe de 16 valeurs provenant soit de la première soit de la seconde moitié de la Color RAM.



-- SYSTEME DE BACKGROUND --

BACKGROUND

Le background est composé de "caractères" qui font 8 pixels de haut et 8 pixels de large. L'écran est organisé en 768 caractères visibles : 32 horizontaux par 24 verticaux.

4 rangées supplémentaires de caractères existent en-dessous des 24 rangées verticales visibles. Ces rangées sont utiles pour scroller de nouvelles données vers le haut de l'écran.

2048 octets de la RAM Vidéo font fonction de map d'écran. Cette map définit les positions à l'écran de 896 caractères (768 visibles).

A chaque position de caractère, il y a un mot de 16 bits (2 octets) qui détermine:

1. Quel caractère parmi les 512 afficher à cet emplacement (9 bits).
2. S'il faut ou pas flipper le caractère horizontalement ou verticalement (2 bits).
3. Lequel des deux sets de 16 couleurs doit être utilisé (1 bit).
4. Si les sprites masquent le background ou inversement (1 bit).
5. Trois bits inutilisés, pouvant être utilisés comme flags (3 bits).

La map d'écran visible occupe 1536 octets de RAM Vidéo (768 caractères, 2 octets par caractère).

Les quatre rangées de caractères invisibles en-dessous de l'écran visible occupent 256 octets (128 caractères de deux octets).

Cela laisse 256 octets de mémoire inutilisée aux plus hautes adresses de la RAM de map d'écran de 2048 octets.
Ces 256 octets sont normalement utilisés pour stocker la Table d'Attributs des Sprites (*Sprite Attribute Table*).

Cette mémoire peut également être utilisée pour stocker 8 patterns de caractères, si la Table d'Attributs des Sprites est localisée ailleurs. Si la map d'écran est placée en $3800 (le cas habituel), les numéros de caractères pour les 256 octets inutilisés de map d'écran RAM vont de $1F0 à $1F7.

BITS D'INHIBITION DE SCROLLING

L'écran de background peut être scrollé par incréments de 1 pixel à la fois horizontalement et verticalement. Deux bits d'"inhibition de scrolling" vous permettent de désactiver le scrolling dans deux zones de l'écran:

Si HSI (*Horizontal Scroll Inhibit*) est mis à 1, la bande horizontale de 2 caractères en haut de l'écran ne scrolle pas.

Si VSI (*Vertical Scroll Inhibit*) est mis à 1, la bande verticale de 8 caractères à droite de l'écran ne scrolle pas.

Cette fonction simplifie le placement d'information de score en haut ou à droite de l'écran. Quand l'écran de background scrolle, le score reste en place à condition que les bits d'inhibition aient été mis à 1.

SPRITES

Un sprite est un objet facilement déplaçable. Le VDP fournit 64 sprites indépendants. Un bit de contrôle détermine la taille de tous les sprites : 0 pour des sprites de 8H x 8V, 1 pour des sprites de 8H x 16V.

Chaque sprite utilise une entrée de 3 octets dans une Table d'Attributs des Sprites (*Sprite Attribute Table*) dans la RAM Vidéo pour définir sa position horizontale et verticale et pour sélectionner un des 256 caractères à afficher.

Un bit de contrôle du VDP (*Sprite Shift*, bit 3 de R0) décale tous les sprites de 8 pixels vers la gauche. Cela permet aux sprites de scroller progressivement au-delà du côté gauche de l'écran.

Les caractères de sprites et de background sont dessinés à partir du même set de patterns de caractères dans la RAM Vidéo. Les sprites ont donc les même capacités de couleur que les caractères de background : 16 couleurs simultanément parmi un choix de 64 couleurs.

Les couleurs des sprites sont toujours prises dans le deuxième groupe de 16 couleurs dans la Color RAM.

Les sprites ne scrollent pas quand le background est scrollé.

Les sprites peuvent être placés au-dessus ou en-dessous d'autres sprites. Les sprites peuvent également apparaître au-dessus ou en-dessous des caractères de background.

Jusqu'à huit sprites peuvent occuper une même ligne d'affichage. Un bit du registre d'état du VDP est fourni pour vous prévenir quand huit sprites ou plus sont positionnés sur la même ligne. Une ligne contenant neuf sprites ou plus n'est pas affichée correctement.

Un autre bit du registre d'état du VDP indique si deux sprites se sont touchés (collision).

TABLE D'ATTRIBUTS DES SPRITES

Une section de 256 octets de la RAM Vidéo fonctionne comme Table d'Attributs des Sprites. Le registre VDP R5 est habituellement mis à $FF pour positionner la Table d'Attributs des Sprites en $3F00, les 256 derniers octets de la RAM Vidéo.

La Table d'Attributs des Sprites est organisée comme l'indique la table sur la page suivante.

Organisation de la Table d'Attributs des Sprites

Adresse	Attribut
3F00	vpos #0
3F01	vpos #1
3F02	vpos #2
3F03	vpos #3
3F04	vpos #4
3F05	vpos #5
3F06	vpos #6
3F07	$D0 (terminaison)
.	
.	
.	
3F3E	
3F3F	vpos #63
3F40	64 octets inutilisés. [Deux caractères de
.	32 octets de numéros $1FA et $1FB peuvent
.	être placés ici]
.	
3F7F	
3F80	hpos #0
3F81	code caract #0
3F82	hpos #1
3F83	code caract #1
3F84	hpos #2
3F85	code caract #2
3F86	
3F87	
.	
.	
3FFE	hpos #63
3FFF	code caract #63

Le code spécial $D0 est placé en code de position verticale pour indiquer au hardware d'arrêter de chercher des sprites dans la liste. Tous les sprites arrivant après l'entrée $D0 sont désactivés.

Quand deux sprites se chevauchent, le sprite ayant le plus grand numéro d'ordre est affiché par dessus l'autre. La priorité de recouvrement est donc déterminée par la position des sprites dans la table.

Les octets de position horizontale localisent le coin supérieur gauche du sprite à l'une des 256 coordonnées horizontales sur l'écran.

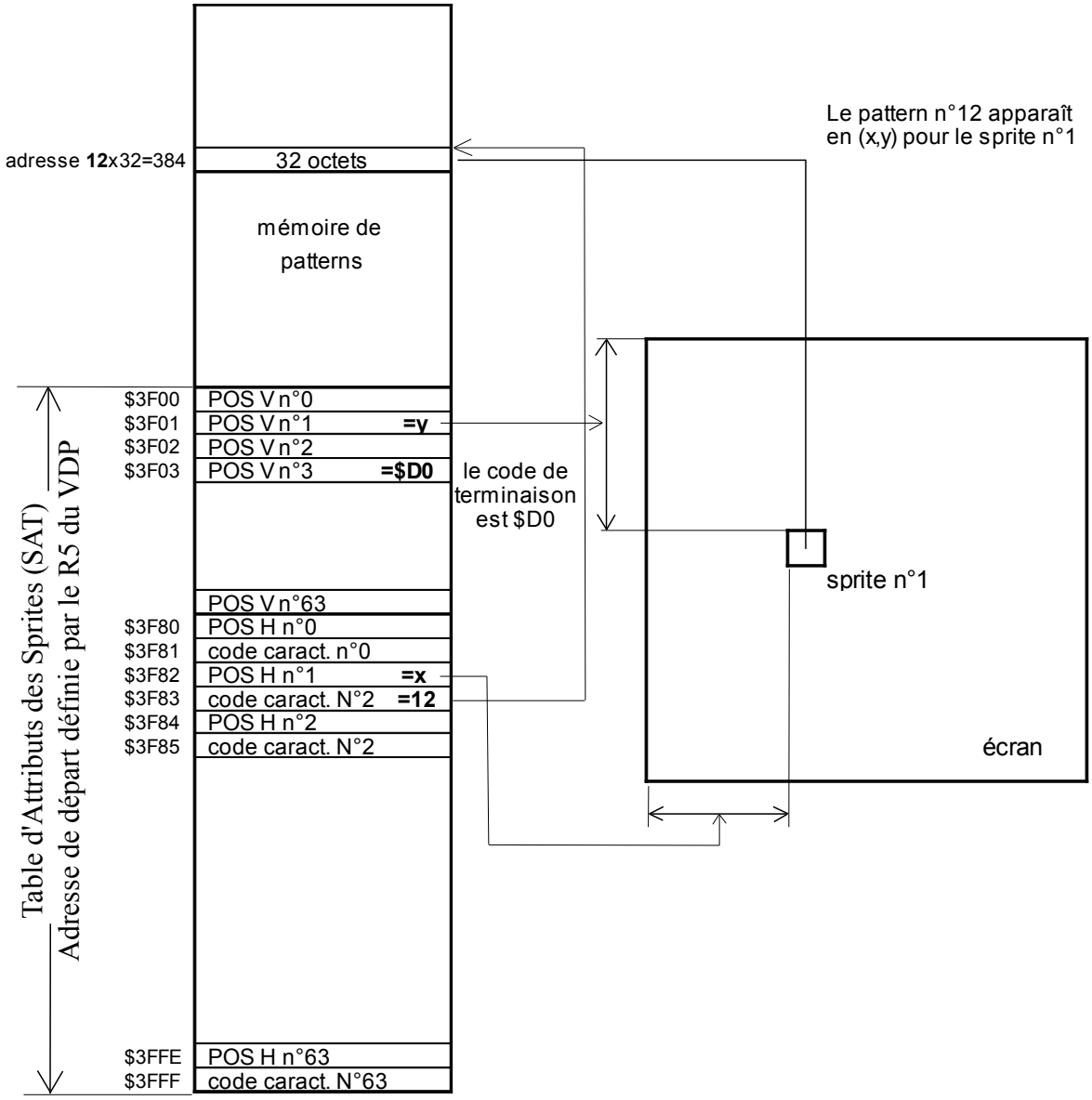
Une valeur hpos=0 met le sprite sur la colonne gauche de l'écran (8 pixels de gauche). hpos=255 met le sprite sur la dernière colonne de pixels; seulement la colonne de pixels de gauche du sprite est affichée et les 7 autres colonnes du sprite sont masquées.

Cela permet de scroller progressivement un sprite vers le côté **droit** de l'écran.

Mettre le *Sprite Shift bit* du VDP (R0, bit 3) à 1 décale tous les sprites de huit pixels vers la gauche. Cela permet de scroller progressivement les sprites vers le côté **gauche** de l'écran.

Si l'apparition et la disparition progressive des sprites est désirée de chaque côté de l'écran, un autre bit de contrôle du VDP (R0, bit 5) peut être mis à 1 pour masquer la colonne de caractères de gauche.

Donc si le bit 0 de R0 est mis à 0 (pas de décalage vers la gauche) et que le bit 5 de R0 est mis à 1 (masquage de la colonne de gauche), l'écran est réduit à 31 colonnes de caractères, et les sprites entrent et sortent de l'écran progressivement. Le côté gauche est géré par le fait que hpos=0 met le sprite dans la colonne de gauche, qui est masquée dans ce mode.



-- Système de Sprites --

PRIORITE D'AFFICHAGE SPRITE/BACKGROUND

Les sprites peuvent apparaître devant ou derrière le background. Ceci est contrôlé par le bit 12 du code de caractère de 16 bits. Si ce bit est mis à 0, tous les sprites apparaissent devant le background. Si ce bit est mis à 1, les couleurs de background n°1 à 15 apparaissent devant les sprites. La couleur de background n°0 apparait toujours derrière les sprites.

COLOR RAM

Le circuit de VDP contient une RAM de couleurs constituée de 32 valeurs de 6 bits. Ces valeurs sont formattées ainsi:

X	X	B1	B0	G1	G0	R1	R0

Où R1:R0, G1:G0 et B1:B0 définissent quatre intensité de rouge, vert et bleu comme suit:

R,G,B-1	R,G,B-0	Intensité
0	0	Off
0	1	1/3
1	0	2/3
1	1	3/3 (le plus lumineux)

La Color RAM est organisée en deux banques de 16 couleurs chacune. Les couleurs sont sélectionnées à l'intérieur de chaque banque par un code couleur de 4 bits.

PATTERNS DE CARACTERES

Les patterns de points destinés aux images à l'écran sont appelés caractères. Les patterns de caractères sont utilisés à la fois par le background et par les sprites.

Chaque caractère est constitué de 32 octets. Ces octets sont organisés en huit groupes de 4 bits, où chaque groupe de 4 octets forme une ligne de huit pixels.

Quatre bits sont nécessaires pour sélectionner 16 couleurs.

La table sur la page suivante détaille un pattern de caractère de 32 octets. Les "0" sont représentés par des points pour plus de clarté.

Un exemple de pattern de caractère

Octet	b7	b6	b5	b4	b3	b2	b1	b0		
0	1	1	1	1	1	1	1	1	c0	PREMIERE LIGNE DE PIXELS:
1	.	.	.	.	.	.	.	.	c1	'0101' = couleur n°5
2	1	1	1	1	1	1	1	1	c2	
3	.	.	.	.	.	.	.	.	c3	
4	.	.	.	.	.	.	.	.	c0	DEUXIEME LIGNE DE PIXELS:
5	.	.	.	.	.	.	.	.	c1	'1100' = couleur n°12,
6	1	1	.	.	.	.	.	.	c2	'0000' = couleur n°0.
7	1	1	.	.	.	.	.	.	c3	
8	.	.	.	.	.	.	.	.	c0	TROISIEME LIGNE DE PIXELS
9	.	.	.	.	.	.	.	.	c1	
10	1	1	.	.	.	.	.	.	c2	
11	1	1	.	.	.	.	.	.	c3	
12	.	.	.	.	.	.	.	.	c0	QUATRIEME LIGNE DE PIXELS
13	.	.	.	.	.	.	.	.	c1	
14	1	1	.	.	.	.	.	.	c2	
15	1	1	.	.	.	.	.	.	c3	
16	.	.	.	.	.	.	.	.	c0	CINQUIEME LIGNE DE PIXELS
17	.	.	.	.	.	.	.	.	c1	
18	1	1	.	.	.	.	.	.	c2	
19	1	1	.	.	.	.	.	.	c3	
20	.	.	.	.	.	.	.	.	c0	SIXIEME LIGNE DE PIXELS
21	.	.	.	.	.	.	.	.	c1	
22	1	1	.	.	.	.	.	.	c2	
23	1	1	.	.	.	.	.	.	c3	
24	.	.	.	.	.	.	.	.	c0	SEPTIEME LIGNE DE PIXELS
25	.	.	.	.	.	.	.	.	c1	
26	1	1	.	.	.	.	.	.	c2	
27	1	1	.	.	.	.	.	.	c3	
28	.	.	.	.	.	.	.	.	c0	HUITIEME LIGNE DE PIXELS
29	1	1	1	1	1	1	1	1	c1	'0010' = couleur n°2
30	.	.	.	.	.	.	.	.	c2	
31	.	.	.	.	.	.	.	.	c3	

Ce pattern représente une lettre "C" multicolore.

La section du haut a la couleur n°5 (lire les bits correspondants du plus significatif au moins significatif: 0101).

La section verticale a la couleur n°12 (1100).

La section du bas a la couleur n°2 (0010).

Les couleurs qu'elles représentent dépendent des valeurs stockées aux emplacement 2, 5 et 12 de la Color RAM.

NOTE: la couleur de fond du caractère 8x8 est la couleur n°0 (0000).

REGISTRES DU VDP

Le VDP est contrôlé par onze registres internes de 8 bits. Cette section décrit la méthode par laquelle le Z80A accède à ces registres, puis détaille la fonction de chaque registre.

Le Z80 "voit" le circuit de VDP par l'intermédiaire de deux ports d'entrée/sortie, $BE et $BF. $BF est le registre de COMMANDE pour une écriture, et le registre de STATUT pour une lecture.

Le port d'entrée/sortie $BE est le registre de lecture/écriture de DONNEES.

Le registre de commande est écrit deux fois successivement pour toutes les opérations de commande; le registre de données peut être lu ou écrit n'importe quel nombre de fois à la suite, selon l'opération.

Le registre de commande étant sensible à la séquence, recquérant une **paire** d'octets par opération, une instruction DI doit précéder les mises à jour de ce registre. Cela empêche, par exemple, qu'une routine d'interruption qui lirait le registre de statut du VDP ne perturbe la synchronisation de deux octets.

Il y a une contrainte de timing concernant l'accès au circuit VDP.

Le circuit de VDP ne peut pas traiter les données plus vite que les vitesses suivantes:

16 T-States du Z80A pendant le VBLANK
29 T-States du Z80A pendant la vidéo active

Cela signifie que vous ne devez jamais envoyer deux instructions OUT ou IN consécutives au VDP; elles doivent être séparées par au moins une instruction NOP.

Cette contrainte s'applique à la RAM Vidéo et à la Color RAM aussi bien qu'aux registres internes.

Un champ de mode de 2 bits constitué des bits 7 et 6 du second octet de COMMANDE détermine une des quatre opérations:

b7	b6	Opération
0	0	Lire dans la RAM de 16Ko du VDP
0	1	Ecrire dans la RAM de 16Ko du VDP
1	0	Ecrire dans un registre du VDP
1	1	Ecrire dans la Color RAM

Pour configurer les deux premières opérations, lecture/écriture dans la RAM Vidéo, les deux octets de COMMANDE ont le format suivant:

Lecture dans la RAM du VDP

Premier octet écrit dans $BF	A7	A6	A5	A4	A3	A2	A1	A0
Deuxième octet écrit dans $BF	0	0	A13	A12	A11	A10	A9	A8

Ecriture dans la RAM du VDP

Premier octet écrit dans $BF	A7	A6	A5	A4	A3	A2	A1	A0
Deuxième octet écrit dans $BF	0	1	A13	A12	A11	A10	A9	A8

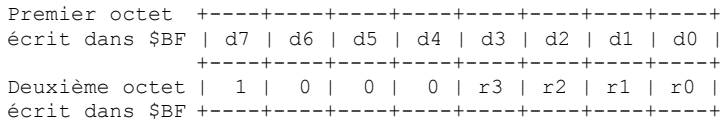
Après que les deux octets de COMMANDE ont été envoyés, les données peuvent être lues sur le port d'entrée $BE ou écrites sur le port de sortie $BE.

Le VDP a une fonction d'auto-incrémentation de l'adresse. Une fois que l'adresse de départ de la RAM Vidéo a été chargée, les données (port d'E/S $BE) peuvent être accédées répétitivement, et l'adresse sera automatiquement incrémentée d'un octet à chaque accès.

NOTE: La fonction d'auto-incrémentation fonctionne uniquement avec seulement des lectures ou seulement des écritures. Changer de lecture vers écriture ou vice-versa recquiert deux écritures dans le registre de commande pour resélectionner le mode.

Le format du registre de COMMANDE pour écrire dans un registre du VDP (ils sont à écriture seule) est montré ci-dessous:

Ecriture dans un Registre du VDP

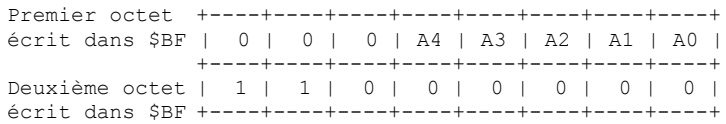


Le numéro de registre est représenté par r3-r0 et varie de 0 à 10 (de $00 à $0A). Les données à écrire sont contenues dans d7-d0.

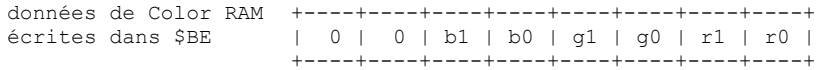
Pour l'opération d'écriture dans le registre seulement, il n'y a pas d'écriture dans le registre de données en $BE, les données (d7-d0) étant contenues dans le premier octet de la paire d'octets de commande.

Le dernier mode de commande permet d'écrire des valeurs de couleur dans la Color RAM à écriture seule:

Ecriture dans la Color RAM



Après avoir chargé cette paire d'octets dans le registre de commande, la Color RAM est écrite sur le port de sortie $BE selon le format suivant:



Où b1-b0 déterminent quatre intensités de bleu, g1-g0 quatre intensités de vert et r1-r0 quatre intensités de rouge.

Comme pour la RAM Vidéo, des écritures répétées sur le port de sortie $BE incrémentent automatiquement l'adresse de Color RAM. Les adresses de Color RAM sont mappées comme indiqué sur la page suivante.

```
Adresse de Color RAM          Numéro de Couleur
-----------------------------------------------
    00000 ($00)               Banque 1, Couleur 0   [Note: Les couleurs
    00001 ($01)               Banque 1, Couleur 1    de background peuvent
    00010 ($02)               Banque 1, Couleur 2    être prises dans l'un
    00011 ($03)               Banque 1, Couleur 3    ou l'autre des deux
    00100 ($04)               Banque 1, Couleur 4    groupes de 16 couleurs]
    00101 ($05)               Banque 1, Couleur 5
    00110 ($06)               Banque 1, Couleur 6
    00111 ($07)               Banque 1, Couleur 7
    01000 ($08)               Banque 1, Couleur 8
    01001 ($09)               Banque 1, Couleur 9
    01010 ($0A)               Banque 1, Couleur 10
    01011 ($0B)               Banque 1, Couleur 11
    01100 ($0C)               Banque 1, Couleur 12
    01101 ($0D)               Banque 1, Couleur 13
    01110 ($0E)               Banque 1, Couleur 14
    01111 ($0F)               Banque 1, Couleur 15

    10000 ($10)               Banque 2, Couleur 0   [Note: les couleurs de
    10001 ($11)               Banque 2, Couleur 1    bordure et les couleurs
    10010 ($12)               Banque 2, Couleur 2    des sprites doivent
    10011 ($13)               Banque 2, Couleur 3    être prises dans le
    10100 ($14)               Banque 2, Couleur 4    deuxième groupe de 16
    10101 ($15)               Banque 2, Couleur 5    couleurs]
    10110 ($16)               Banque 2, Couleur 6
    10111 ($17)               Banque 2, Couleur 7
    11000 ($18)               Banque 2, Couleur 8
    11001 ($19)               Banque 2, Couleur 9
    11010 ($1A)               Banque 2, Couleur 10
    11011 ($1B)               Banque 2, Couleur 11
    11100 ($1C)               Banque 2, Couleur 12
    11101 ($1D)               Banque 2, Couleur 13
    11110 ($1E)               Banque 2, Couleur 14
    11111 ($1F)               Banque 2, Couleur 15
```

Le registre de commande a une fonction supplémentaire. Quand il est lu, avec une instruction IN ($BF), il fonctionne comme registre de statut d'interruption, et désactive les requêtes d'interruption provenant du circuit VDP.

MISES A JOUR DES REGISTRES DU VDP

En général, toutes les mises à jour du VDP doivent être effectuées pendant que le signal vidéo TV est désactivé (*blanked*). Si des opérations affectant l'écran ne sont pas faites pendant le blanking de l'écran, du “bruit” apparaitra sur l'image.

Votre programme peut détecter les intervalles de blanking horizontal et vertical en utilisant le système d'interruption et les bits de flag du VDP. Cette possibilité est discutée dans la section CPU.

Il n'est pas nécessaire de synchroniser l'initialisation du VDP avec les intervalles de blanking, l'écran pouvant être désactivé avant les opérations d'initialisation, puis activé quand l'initialisation est terminée.

DESCRIPTION DES REGISTRES DU VDP

Registres 0 et 1

R0 et R1 définissent le mode d'affichage et le mode d'interruption du VDP .

```
R0:   b7 VSI Vertical Scroll Inhibit (8 colonnes de caractères de droite).
           (Inhibition du Scrolling Vertical)
           0: Scrolle avec le background.
           1: Fixe (pas de scrolling).

      b6 HSI Horizontal Scroll Inhibit (2 lignes de caractères du haut).
           (Inhibition du Scrolling Horizontal)
           0: Scrolle avec le background.
           1: Fixe (pas de scrolling).

      b5 LCB Left Column Blank. (Masquage de la colonne de gauche)
           0: Affichage normal.
           1: Masque la colonne de caractères de gauche (couleur de bordure).

      b4 IE1 Interrupt Enable. (Interruption activée)

      b3 SS  Sprite Shift. (Décalage des sprites)
           0: Position normale des sprites.
           1: Décale tous les sprites de 8 pixels vers la gauche.

      b2 1   [Toujours mettre 1].

      b1 1   [Toujours mettre 1]

      b0 ES  External Sync.  (Synchro. Externe)
           [Toujours mettre 0].

R1:   b7 1   [Toujours mettre 1]

      b6 DIS Display ON. (Affichage activé)
           0: Désactiver l'écran.
           1: Vidéo normale.

      b5 IE  Interrupt Enable. (Interruption activée)

      b4 0   [Toujours mettre 0]

      b3 0   [Toujours mettre 0]

      b2 0   [Toujours mettre 0]

      b1 SZ  Sprite Size. (Taille des sprites)
           0: Tous les sprites font 8 par 8.
           1: Tous les sprites font 8 par 16.

      b0 0   [Toujours mettre 0].
```

Le champ de contrôle d'interruption sur 2 bits est défini comme suit:

```
      IE IE1      Interruption activée
      -- ---      --------------------
       0  1       Ligne de raster.
       1  1       VBLANK et Ligne de raster.
```

Registre 2

R2 définit l'adresse de base de la map d'écran. Il positionne la map d'écran de 2048 octets à l'une des huit adresses, alignées sur des blocs de 2048 octets, selon la table suivante:

Valeur de R2	Adresse de base de la map d'écran
$FF	$3800 <--- Configuration Normale
$FD	$3000
$FB	$2800
$F9	$2000
$F7	$1800
$F5	$1000
$F3	$0800
$F1	$0000

La configuration normale pour R2 est $FF, qui positionne l'adresse de base en $3800, la section de 2 Ko la plus haute de la RAM Vidéo de 16 Ko.

Registre 3

R3 doit toujours être initialisé à $FF.

Registre 4

R4 doit toujours être initialisé à $FF.

Registre 5

R5 contrôle l'adresse de base de la Table d'Attributs des Sprites. Cette table de 256 octets peut être positionnée à une parmi 64 adresses de départ dans la RAM Vidéo.

L'adresse de base est formattée dans R5 comme suit:

```
        +---+-----+-----+-----+-----+----+----+---+
R5:     | 1 | A13 | A12 | A11 | A10 | A9 | A8 | 1 |
        +---+-----+-----+-----+-----+----+----+---+
```

La configuration normale pour R5 est $FF, qui positionne la Table d' Attributs des Sprites en $3F00. Quand R2 vaut $FF, plaçant la map d'écran en $3800, et que R5 vaut $FF, les 256 octets du haut de la map d'écran inutilisés sont occupés par la Table d'Attributs des Sprites de 256 octets.

Registre 6

R6 définit l'adresse de base des patterns de sprites. Seulement deux valeurs sont valides pour R6:

Valeur de R6	Patterns de sprites	
$FB	Premier bloc de 8 Ko de la RAM Vidéo	<--- Valeur Normale
$FF	Deuxième bloc de 8 Ko de la RAM Vidéo	

L'emplacement privilégié pour les patterns de sprites est dans le premier bloc de 8 Ko de la RAM Vidéo, car la totalité des 8192 octets est disponible pour les patterns. Dans le deuxième bloc de 8 Ko, 2048 octets sont perdus pour la map d'écran et la Table d'Attributs des Sprites. Donc 256 patterns de sprites peuvent être stockés dans le premier bloc de 8 Ko, mais seulement 192 dans le deuxième bloc de 8 Ko.

Registre 7

R7 définit la couleur de bordure. Cette couleur est prise dans la deuxième banque de couleurs de la Color RAM du VDP.

R7:	1	1	1	1	C3	C2	C1	C0

Registre 8

R8 définit la valeur de scrolling horizontal du background. Une valeur de $00 ne produit pas de scrolling. Une valeur de $01 déplace le background d'un pixel vers la gauche, et déplace la colonne de pixels la plus à gauche vers la colonne la plus à droite (le scrolling “boucle”).

De plus grandes valeurs de R8 effectuent une “rotation” horizontale de l'écran jusqu'à un maximum de 255 pixels pour une valeur de $FF. Notez que le scrolling horizontal ressemble à un cylindre tournant selon l'axe vertical, l'information graphique disparaissant à gauche et réapparaissant à droite de l'écran.

Registre 9

R9 définit la valeur de scrolling vertical du background. Une valeur de $00 ne produit pas de scrolling. Une valeur de $01 déplace le background d'un pixel vers le haut. Le scrolling vertical ressemble à un cylindre tournant suivant l'axe horizontal, cependant le point de “bouclage” n'est pas au bord de l'écran, comme pour le scrolling horizontal. En fait, il s'agit de la 28ème rangée de caractères. (Souvenez-vous que l'écran est organisé en 28 lignes de caractères, avec les 24 lignes du haut affichées à l'écran).

Le registre 9 est *latché* (verrouillé) pendant le blanking vertical.

Registre 10

R10 contrôle l'interruption de ligne de raster.

Le rayon de la TV balaie l'écran suivant des lignes horizontales (appelées lignes de raster) de la gauche vers la droite, se déplaçant du haut vers le bas de l'écran. L'image visible contient 192 de ces lignes de raster.

A la fin de chaque ligne de raster il y a un bref intervalle de blanking (HBLANK) durant lequel le rayon de la TV est éteint tandis qu'il revient du côté droit vers le côté gauche de l'écran. Cet intervalle de HBLANK dure approximativement 10 microsecondes.

On peut désirer recevoir une interruption juste après qu'une certaine ligne de raster ait été affichée. Par exemple, vous pourriez vouloir changer la valeur d'une couleur après que le rayon ait balayé la moitié supérieure de l'écran. Dans ce cas vous voudriez savoir quand la 95ème ligne a été complétée.

R10 vous permet de réaliser cela. Si vous chargez 94 dans R10, une interruption de Ligne de Raster sera générée chaque fois que la 95ème ligne aura fini d'être affichée.

La valeur chargée dans R10 doit être le numéro de ligne auquel vous désirez déclencher l'interruption, diminué de 1.

L'interruption de ligne de raster continue de générer des requêtes d'interruption tant que le rayon balaie l'écran. La table suivante indique les valeurs de R10 et les lignes de raster qui génèrent les requêtes d'interruption au moment des HBLANK:

Valeur de R10	**Requêtes d'interruption sur ces HBLANKS**
$C0-$FF	Aucune
$00	1,2,3,4,5,...191
$01	2,4,6,8,10,..190
$02	3,6,9,12,....189
$03	4,8,12,16,...188
(etc)	(etc)

Deux cas particuliers: $FF désactive les requêtes d'interruption, et $00 donne une requête d'interruption de ligne de raster pour chaque ligne horizontale.

Il y a un délai d'une interruption entre le chargement de R10 et le moment où la valeur prend effet. Par exemple, si vous chargez R10 pour la ligne 20, et répondez à l'interruption en réinitialisant R10 à 150, la prochaine interruption de ligne de raster interviendra à 20 et toutes les suivantes à 150.

Les registres du VDP sont initialisés à la mise sous tension avec les valeurs suivantes:

```
R0     $36    ; Mode
R1     $A0    ; Mode
R2     $FF    ; Base de la map d'écran
R3     $FF    ; (Toujours $FF)
R4     $FF    ; (Toujours $FF)
R5     $FF    ; Base de la Table des Sprites
R6     $FB    ; Base de la Table des Patterns de sprites
R7     $00    ; Couleur de bordure n°0
R8     $00    ; Scroll-H
R9     $00    ; Scroll-V
R10    $FF    ; Interruption de ligne de raster($FF=off)
```

GESTION DE LA MEMOIRE

La Mk3 contient 8 Kilooctets de ROM et 8 Kilooctets de RAM. Les ROM de programme se connectent dans la machine en utilisant un des deux slots disponibles.

Le slot de façade accepte une carte ressemblant à une carte de crédit. La capacité de cette carte est de 32 Kilooctets. Une carte de 128 Kilooctets est en préparation.

Le slot du dessus accepte une cartouche plus conventionnelle. Des cartouches de 128 Kilooctets sont actuellement en production. Des cartouches de 256 Kilooctets et plus sont en préparation.

Le processeur Z80A est capable d'adresser 64 Kilooctets de mémoire. La mémoire de la cartouche de jeu pouvant excéder 64 Kilooctets, un système de gestion de mémoire simple est implémenté dans la Mk3.

Quand la Mk3 est allumée, la map mémoire contient une ROM de 8 Kilooctets en $0000, et une RAM de 8 Kilooctets en $C000.

RAM DU SYSTEME

La RAM est mappée en deux endroits, $C000-$DFFF et $E000-$FFFF. Cette RAM est utilisée pour la pile du Z80A et comme zone de travail.

Les huit emplacements au sommet de la RAM sont réservés pour les registres de contrôle de la gestion de la mémoire, et ne doivent pas être utilisés par un programme de jeu.

Pour la compatibilité future, la RAM doit être adressée dans l'intervalle $C000-$DFFF. La pile doit être placée en $DFF8.

ACTIVATION DE MEMOIRE

Le port de sortie $3E contrôle l'activation de mémoire. Des bits individuels de ce port activent la ROM système, la RAM système, le slot de carte, le slot de cartouche, et le slot externe. (Le slot externe n'est pas utilisé pour l'instant).

Le port de sortie $3E a le format suivant:

```
OUT ($3E):
----------
bit 0    sans importance

bit 1    sans importance

bit 2    0: active les joysticks
         1: désactive les joysticks

bit 3    0: active les 8Ko de ROM de la machine
         1: désactive les 8Ko de ROM de la machine

bit 4    0: active les 8Ko de RAM interne
         1: désactive les 8Ko de RAM interne

bit 5    0: active la ROM de la carte
         1: désactive la ROM de la carte

bit 6    0: active la ROM de la cartouche
         1: désactive la ROM de la cartouche

bit 7    0: active le port externe
         1: désactive le port externe
```

SELECTION DE MEMOIRE A LA MISE SOUS TENSION

Quand le système est mis sous tension, le Z80A commence l'exécution du code à l'adresse $0000 dans la ROM résidente de 8 Kilooctets. Ce code en ROM réalise les étapes suivantes:

1. Le système est initialisé.

2. Un petit programme est copié de la ROM vers la RAM (en $C700).

3. Le programme saute à l'adresse $C700 pour exécuter ce programme.

4. Tous les emplacements de ROM du système sont désactivés. C'est pourquoi le programme est copié et tourne en RAM – les emplacements de ROM peuvent alors être testés un par un.

5. Les emplacements de ROM sont activés individuellement et la présence de mémoire est vérifiée. Ils sont testés dans l'ordre suivant:
   A. Slot de carte de façade.
   B. Slot de cartouche du dessus.
   C. Slot externe (à l'arrière).

6. Si une mémoire est détectée dans n'importe lequel de ces slots, le slot est activé et le Z80A exécute un saut à l'adresse $0000.

7. Si aucune mémoire n'est détectée, la ROM de la console est activée et un message “Insérez une cartouche” est affiché à l'écran.

Notez que si la carte et la cartouche sont connectées en même temps, la carte sera activée car elle est testée en premier.

Un programme de jeu est écrit comme une application Z80A entièrement autonome ayant pour origine l'adresse $0000. Un programme de jeu doit s'occuper de tout le “ménage”, par exemple la configuration des vecteurs d'interruptions.

Une fois qu'un jeu tourne, il n'y a aucun accès à la boot ROM.

CARTES MEMOIRE

Les schémas de la page suivante montrent les cartes mémoire pour deux tailles mémoire définies: 32 Kilooctets et 128 Kilooctets.

Le schéma M-1 montre la carte mémoire du Z80A au démarrage.

32 KILOOCTETS

Le schéma M-2 montre la carte mémoire du Z80A pour la carte de 32 Kilooctets. C'est le cas le plus simple: 32 Kilooctets commençant à l'adresse $0000.

128 KILOOCTETS

Le schéma M-3 montre la carte mémoire pour une cartouche de 128 Kilooctets. (On y fait référence dans la littérature Sega en tant que cartouche “Mega”(bit)).

Comme avec la mémoire de 32 Kilooctets, il y a une section de ROM contigüe dans la moitié inférieure de la mémoire du Z80A. Cette section est toujours disponible pour le Z80A.

De plus, il y a six banques de ROM de 16 Kilooctets aux adresses mémoire $8000-$BFFF. Une seule de ces banques est disponible pour le Z80A à un instant donné. A partir du moment où une banque est sélectionnée, les cinq autres sont inactives.

Un registre situé à l'adresse $FFFF contrôle laquelle de ces six banques est connectée à l'espace mémoire du Z80A.

Le Registre de Contrôle de Banque suit le format suivant:

```
$FFFF       0 0 0 0 0 b2 b1 b0
```

Les banques sont numérotées de 2 à 7 (b2:b1:b0).

Bien que ce registre à écriture seule existe dans la cartouche mémoire, les écritures dans ce registres sont dupliquées en RAM à l'adresse $FFFF. Cela signifie que le registre de sélection de banque parait être en lecture/écriture, alors que ce qui est réellement lu est l'image du registre en RAM, plutôt que le contenu du registre lui-même.

CARTE MEMOIRE

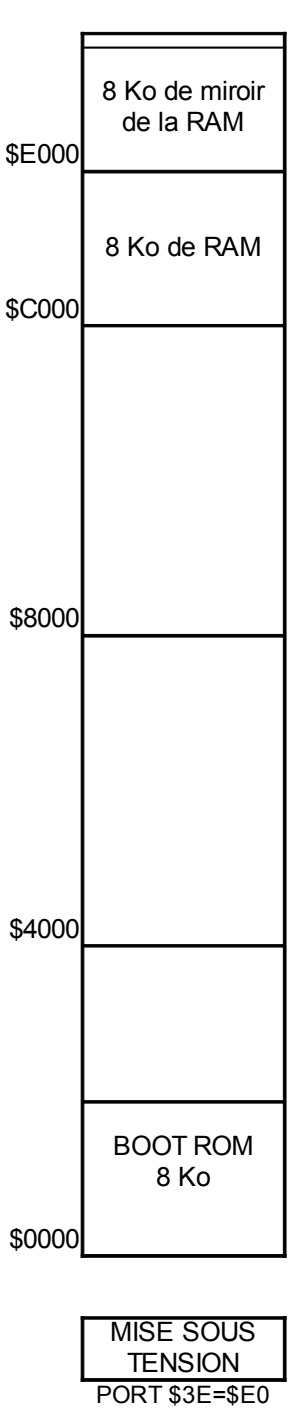


Figure M-1

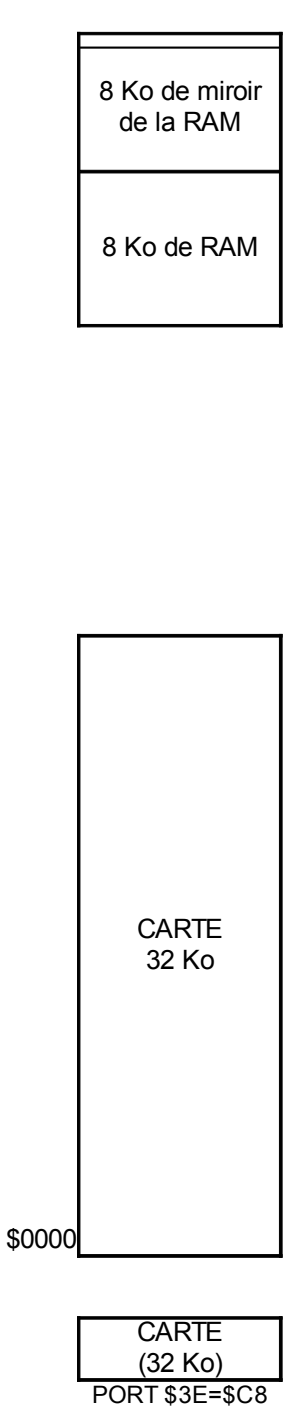
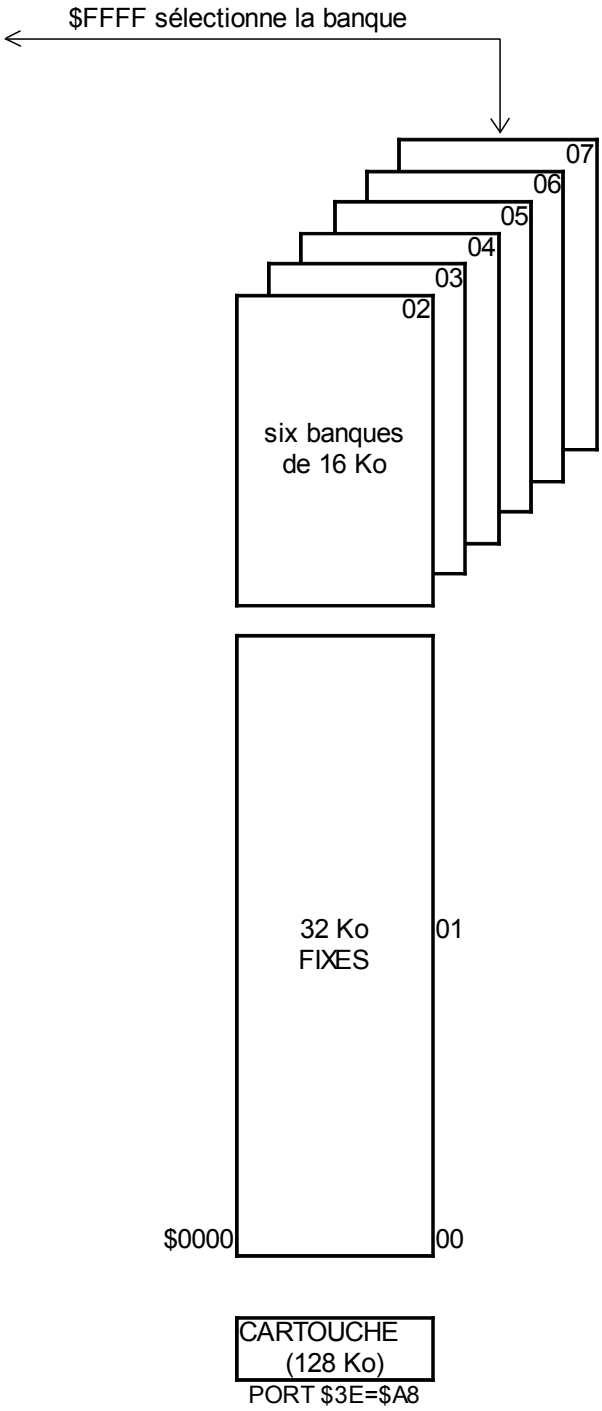


Figure M-2



1 Megabit
Figure M-3

LE HEADER DE CARTOUCHE

La présence d'une cartouche est testée en vérifiant des informations spécifiques en ROM. Ces informations doivent être correctes pour que la cartouche soit reconnue.

Normalement, Sega ajoutera les informations correctes à la cartouche avant la production. Les informations suivantes sont seulement données pour référence.

Le format du *header* (en-tête) est le suivant (adresses indiquées pour les cartouches de 32 Ko et plus):

En $7FF0:

La chaîne "TMR SEGA" [54,4D,52,20,53,45,47,41,20,20]

En $7FFA-$7FFB

Une *checksum* (somme de contrôle) de la ROM (dans l'ordre poids faible-poids fort).
$7FFA et $7FFB ne sont pas incluses dans cette checksum.

En $7FFC-$7FFD

Un numéro de série de 16 bits, assigné par Sega.

En $7FFE

Un numéro de révision de logiciel.

En $7FFF

Un octet codant la taille de la ROM:

```
$4A    8K     (header en $1FF0)
$4B    16K    (header en $3FF0)
$4C    32K
$4D    48K
$4E    64K
$4F    128K (1M)
$40    256K (2M)
```

PORTS D'ENTREE/SORTIE

Huit ports d'entrée/sortie sont utilisés par la Mk3:

Port	Direction	Fonction
$3E	sortie	Activation Mémoire
$3F	sortie	Port de contrôle Joystick
$7E	entrée	Spot du Light Gun (vertical)
$7F	entrée	Spot du Light Gun (horizontal)
	sortie	*Programmable Sound Generator* (PSG, Générateur de son programmable)
$BE	e/s	Registre de Données du VDP
$BF	e/s	Registre de Commande/Statut du VDP
$DC	entrée	Port Joystick
$DD	entrée	Port Joystick

Le port $3E est décrit dans la section GESTION MEMOIRE.

Le port $7F est décrit dans la section PSG.

Les ports $BE et $BF sont décrits dans la section VDP.

Les ports restants ont trait aux contrôleurs reliés à la machine, et sont décrits dans cette section.

Les Ports $DC-$DD sont connectés aux deux connecteurs à 9 broches situés en façade de la console.
Bien que ces ports soient appelés “Ports Joystick”, il y a en réalité trois périphériques qui peuvent être connectés en façade:

1. Les joysticks, qui incluent un stick et deux boutons.
2. Le Light Gun.
3. Un “Sports Pad” qui consiste en une trackball et deux boutons.

ATTRIBUTION DES BITS DES PORTS D'ENTREE

Les bits des ports d'entrée $DC et $DD sont utilisés pour les Joysticks, le Light Gun et le Sports Pad (trackball). Ces deux ports fonctionnent en conjonction avec le port $3F, qui définit la direction de quatre des bits, et fournit deux bits de *strobe* (impulsion) pour le Sports Pad. La table ci-dessous montre comment les trois ports sont interprétés pour les options de Joystick et de Gun. Tous les bits indiqués sont des entrées.

Port	Bit	Joystick	Gun
$DC	0	J1-haut	---
$DC	1	J1-bas	---
$DC	2	J1-gauche	---
$DC	3	J1-droite	---
$DC	4	J1-bouton 1	Gâchette J1
$DC	5	J1-bouton 2	---
$DC	6	J2-haut	---
$DC	7	J2-bas	---
$DD	0	J2-gauche	---
$DD	1	J2-droite	---
$DD	2	J2-bouton 1	Gâchette J2
$DD	3	J2-bouton 2	---
$DD	4	RESET	RESET
$DD	5	---	---
$DD	6	---	Impulsion lumineuse J1
$DD	7	---	Impulsion lumineuse J2
$3F	0	1	1
$3F	1	1	Activation latch J1
$3F	2	1	1
$3F	3	1	Activation latch J2
$3F	4	1	1
$3F	5	1	1
$3F	6	1	1
$3F	7	1	1

COMMENT FONCTIONNE LE GUN

Le Gun contient une lentille et une cellule photosensible qui “voit” un spot sur l'écran de la TV. Ce spot est circulaire, et son diamètre augmente lorsque la distance entre le gun et l'écran augmente.

Quand le rayon de la TV atteint la zone pointée par le Gun, deux choses se passent:

1. Une impulsion négative d'environ 20-30 microsecondes est générée sur le bit 6 du port $DD (Joueur 1) ou le bit 7 du port $DD (Joueur 2). Cette impulsion est assez longue pour qu'une boucle logicielle la reconnaisse. La largeur de cette impulsion varie selon la configuration du jeu et le type de TV.

2. La position du rayon est *latchée* (verrouillée) dans deux registres, qui sont lus sur les ports d'entrée $7E (position verticale) et $7F (position horizontale).

La position du rayon peut être latchée soit par un gun branché sur le connecteur du joueur 1, soit par un gun branché sur le connecteur du joueur 2, soit par les deux. [NOTE: Le connecteur du joueur 1 est celui de gauche quand on regarde la console de face].

Deux bits du port de sortie $3F déterminent quel gun latche la position du rayon. Ils sont notés “Activation latch J1” et “Activation latch J2” dans la table précédente.

La routine de lecture du gun exécute habituellement les étapes suivantes:

1. Le switch de gâchette est vérifié. Normalement le gun n'est pas vérifié jusqu'à ce que quelqu'un presse la gâchette. (Les données sont cependant continuellement disponibles).

2. Quand la gâchette est pressée, attente du prochain VBLANK.

3. Lors du VBLANK, affichage d'un écran tout blanc (tous les octets de couleurs valent $3F). Ceci fournit suffisament de lumière sur l'écran pour que le gun soit lu quelle que soit sa position.

4. Pendant le prochain scan, recherche continue d'une transition haut-bas du bit d'impulsion lumineuse. (L'impulsion lumineuse du J1 est présente sur le bit 6 du port $DD; celle du J2 est sur le bit 7 du port $DD).

5. Quand l'impulsion est détectée, lecture du Registre de Position Horizontale en $7F et du Registre de Position Verticale en $7E.

6. Attente d'une transition bas-haut du bit d'impulsion lumineuse. Répétition de l'étape 5 pour toutes les lignes horizontales actives.

7. Quand un scan est terminé, restauration des valeurs normales des couleurs.

Le Gun réagit à un spot circulaire sur l'écran de la TV, pas à un pixel unique. De la manière dont le spot circulaire est “vu” par la cellule photosensible du Gun, des impulsions négatives répétées se produiront pendant plusieurs lignes de scan consécutives.

Les valeurs horizontales du cercle détecté vont, tandis que le registre de position verticale sera incrémenté, diminuer, augmenter, puis cesser lorsque le bord gauche du cercle scanné sera lu par la lentille du gun.

C'est pourquoi il est conseillé d'effectuer une sorte de moyenne et d'extrapolation pour parvenir à une position centrale du cercle scanné.

COMMENT LA TRACKBALL FONCTIONNE

L'interface avec la trackball consiste en six bits d'entrée et un bit de sortie.

Trackball	direction	Joueur 1	Joueur 2
b0	entrée	$DC bit 0	$DC bit 6
b1	entrée	$DC bit 1	$DC bit 7
b2	entrée	$DC bit 2	$DD bit 0
b3	entrée	$DC bit 3	$DD bit 1
S1	entrée	$DC bit 4	$DD bit 2
S2	entrée	$DC bit 5	$DD bit 3
STROBE	sortie	$3F bit 5	$3F bit 7

Avant d'utiliser la trackball, les instructions suivantes doivent être exécutées pour configurer le bit de STROBE en sortie, et initialiser les signaux de strobe à 1.

NOTE: ces instructions sont exécutées lors de l'initialisation de mise sous tension.

Le signal de strobe est alors utilisé pour synchroniser en entrée 4 *nibbles* (groupes de 4 bits) b3-b0, qui apparaissent dans l'ordre suivant:

Nibble n°	Données (b3 b2 b1 b0)	Quand le STROBE est:
1	X7 X6 X5 X4	BAS
2	X3 X2 X1 X0	HAUT
3	Y7 Y6 Y5 Y4	BAS
4	Y3 Y2 Y1 Y0	HAUT

Les positions X,Y de la trackball sont représentées par X7-X0 et Y7-Y0. Ce sont des compteurs 8 bits qui bouclent de $FF à $00.

Quand la balle est envoyée à droite, la valeur de X augmente; quand elle est envoyée à gauche, elle diminue. Quand la balle est envoyée en haut, la valeur de Y augmente; quand elle est envoyée en bas, elle diminue.

Le timing des transitions du STROBE est important. Voici la séquence requise:

Séquence pour lire la Trackball

[Commencer avec le STROBE HAUT].

1. STROBE BAS.
2. Attendre 80 microsecondes.
3. Lire le premier nibble.
4. STROBE HAUT.
5. Attendre 40 microsecondes.
6. Lire le second nibble.
7. STROBE BAS.
8. Attendre 40 microsecondes.
9. Lire le troisième nibble.
10. STROBE HAUT.
11. Attendre 40 microsecondes.
12. Lire le quatrième nibble.
13. (fini-le STROBE est HAUT).

L'annexe D donne le code pour lire la trackball à chaque interruption.

GENERATEUR DE SON PROGRAMMABLE (PSG)

Le PSG (Programmable Sound Generator) dispose de quatre canaux de son, composés de trois générateurs tonals et d'un générateur de bruit.
Chaque canal a son contrôle de volume indépendant (atténuateur). Le PSG est contrôlé via le port $7F.

FREQUENCE DES GENERATEURS TONALS

La fréquence (hauteur) de chaque générateur tonal est fixée par une valeur 10 bits. Cette valeur est décomptée jusqu'à ce qu'elle atteigne zéro: à ce moment la sortie bascule et la valeur 10 bits est rechargée dans le compteur. Par conséquent, les plus grandes valeurs produisent les fréquences les plus basses.

Pour charger une nouvelle valeur de fréquence dans un des trois générateurs, vous devez écrire une paire d'octets sur le port d'entrée/sortie $7F suivant le format suivant:

```
                +---+----+----+----+----+----+----+----+
Premier octet:  | 1 | R2 | R1 | R0 | d3 | d2 | d1 | d0 |
                +---+----+----+----+----+----+----+----+
Deuxième octet: | 0 | 0  | d9 | d8 | d7 | d6 | d5 | d4 |
                +---+----+----+----+----+----+----+----+
```

Le champ R2:R1:R0 sélectionne le canal tonal comme suit:

R2	R1	R0	Canal tonal
0	0	0	n°1
0	1	0	n°2
1	0	0	n°3

La donnée 10 bits est (msb) d9 d8 d7 d6 d5 d4 d3 d2 d1 d0 (lsb).

(msb=bit de poids le plus fort, lsb=bit de poids le plus faible).

CONTROLE DU GENERATEUR DE BRUIT

Le générateur de bruit utilise trois bits de contrôle pour sélectionner le “caractère” du bruit. Un bit appelé "FB" (Feedback) produit soit un bruit périodique soit un bruit “blanc”:

FB	Type de bruit
0	Périodique (comme un signal tonal basse fréquence)
1	Blanc (*hiss*)

La fréquence du bruit est sélectionnée par deux bits NF1:NF0 selon cette table:

NF1	NF0	Horloge-source du générateur de bruit
0	0	Clock/2 [pitch le plus haut]
0	1	Clock/4
1	0	Clock/8 [pitch le plus bas]
1	1	Générateur tonal n°3

Note: “Clock” (le signal d'horloge) a une fréquence fixe. C'est un oscillateur contrôlé par un crystal connecté au PSG.

Quand NF1:NF0 vaut 11, le générateur tonal n°3 fournit l'horloge-source du bruit. Cela permet de “sweeper” (balayer) la fréquence du bruit. Cet effet peut servir à simuler le réacteur d'un jet, par exemple.

Pour charger ces bits de contrôle du générateur de bruit, écrivez l'octet suivant sur le port d'E/S $7F:

	7	6	5	4	3	2	1	0
Sortie($7F):	1	1	1	0	0	FB	NF1	NF0

ATTENUATEURS

Quatre atténuateurs ajustent le volume des trois générateurs tonals et du canal de bruit. Quatre bits A3:A2:A1:A0 contrôlent l'atténuation comme suit:

A3	A2	A1	A0	Attenuation
0	0	0	0	0 db (volume maximum)
0	0	0	1	2 db
0	0	1	0	4 db
0	0	1	1	6 db
0	1	0	0	8 db
0	1	0	1	10 db
0	1	1	0	12 db
0	1	1	1	14 db
1	0	0	0	16 db
1	0	0	1	18 db
1	0	1	0	20 db
1	0	1	1	22 db
1	1	0	0	24 db
1	1	0	1	26 db
1	1	1	0	28 db
1	1	1	1	-Off-

NOTE: d'une plus grande atténuation résulte un son plus faible.

Les atténuateurs pour chaque canal se configurent en écrivant les octets suivants sur le port $7F:

Générateur tonal n°1:	1	0	0	1	A3	A2	A1	A0
Générateur tonal n°2:	1	0	1	1	A3	A2	A1	A0
Générateur tonal n°3:	1	1	0	1	A3	A2	A1	A0
Générateur de bruit:	1	1	1	1	A3	A2	A1	A0

EXEMPLE

Quand la Mk3 est mise sous tension, le code suivant est exécuté:

```
      LD    HL,CLRTB     ; clear table
      LD    C,PSG_PRT    ; le port psg est $7F
      LD    B,4          ; charge quatre octets
      OTIR               ; écriture
      (etc.)

CLRTB defb  $9F,$BF,$DF,$FF
```

Ce code coupe le son des quatre canaux. C'est une bonne idée d'exécuter ce code au moment où le bouton PAUSE est appuyé, afin que le son ne reste pas continuellement pendant la pause.

ANNEXE A – REGISTRES DU VDP

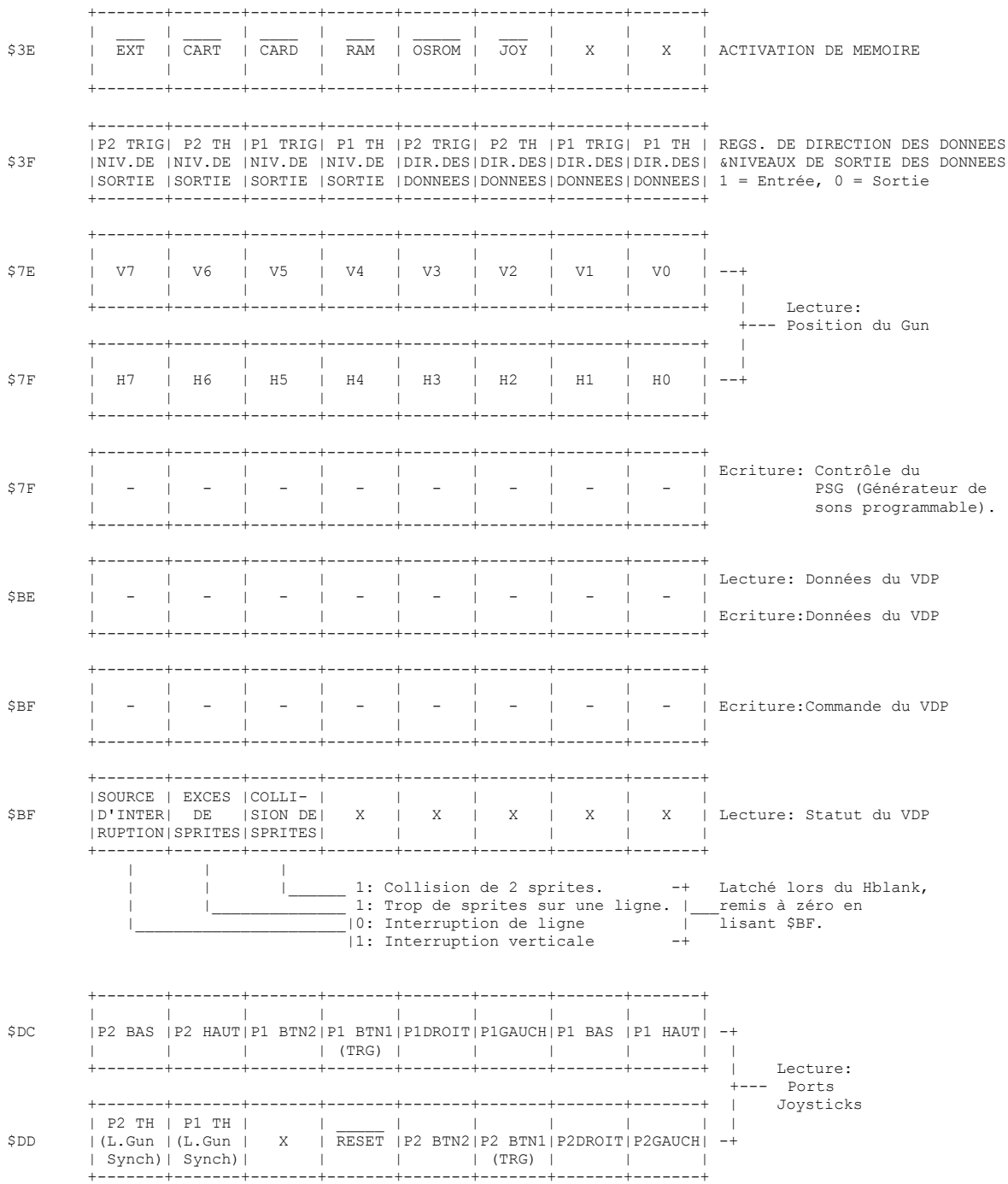
	7	6	5	4	3	2	1	0		
R0	Bit d'Inhib° Vscroll	Bit d'Inhib° HScroll	Masqua-ge col. GAUCHE	IE1	Bit de décalag Sprites	1	1 (M3)	0		
R1	1	Bit d'activ° écran	IE	0 (M1)	0 (M2)	0	Taille Sprites 8 ou 16	Bit de Magnif. Sprites		
R2	1	1	1	1	BASE ECRAN 2	BASE ECRAN 1	BASE ECRAN 0	1	$FF	(Base d'écran en $3800)
R3	1	1	1	1	1	1	1	1	$FF	
R4	1	1	1	1	1	1	1	1	$FF	
R5	1	BASE SAT 5	BASE SAT 4	BASE SAT 3	BASE SAT 2	BASE SAT 1	BASE SAT 0	1	$FF	(SAT en $3F00) SAT = Table d'Attributs des Sprites
R6	1	1	1	1	1	BANQUE CGEN 0	1	1	$FB	(CGEN en $0000) CGEN=Générateur de caractères
R7	1	1	1	1	BORDURE 3	BORDURE 2	BORDURE 1	BORDURE 0		Couleur de bordure
R8	HSCROLL 7	HSCROLL 6	HSCROLL 5	HSCROLL 4	HSCROLL 3	HSCROLL 2	HSCROLL 1	HSCROLL 0	$00	Scrolling Horizontal
R9	VSCROLL 7	VSCROLL 6	VSCROLL 5	VSCROLL 4	VSCROLL 3	VSCROLL 2	VSCROLL 1	VSCROLL 0	$00	Scrolling Vertical
R10	HLI 7	HLI 6	HLI 5	HLI 4	HLI 3	HLI 2	HLI 1	HLI 0	$FF	Interruption de ligne de raster ($C1-$FF=Désactivée)

Lecture du port $BF (statut VDP)	7	6	5	4	3	2	1	0
	SOURCE D'INTERRUPTION	EXCES DE SPRITES	COLLISION DE SPRITES	X	X	X	X	X

Ecriture sur le port $BF (contrôle du VDP) r3:r2:r1:r0=n°de registre		7	6	5	4	3	2	1	0
	Premier octet:	Données à écrire							
	Deuxième octet:	1	0	X	X	r3	r2	r1	r0

NOTE: Désactiver auparavant les interruptions car il faut toujours écrire deux fois dans le registre de commande du VDP.
(les interruptions lisent le registre de statut et les accès aux registres du VDP sont sensibles à la séquence)

ANNEXE B – PORTS D'ENTREE/SORTIE



ANNEXE C – EXTRAIT DE CODE DE LECTURE DU GUN

```
 1:
 2:
 3:   CDATA   EQU     0C100H          ; Gun Status Work (1)
 4:   HVCNT   EQU     CDATA+1         ; RAM Save Counter (1)
 5:   HVDATA  EQU     HVCNT+1         ; V.H Counter Save Work (40H)
 6:   GCHKCT  EQU     HVDATA+040H     ; (1)
 7:   GCHKWK  EQU     GCHKCT+1        ; (10)
 8:   SHOOTF  EQU     GCHKWK+10       ; Gun Shot Flag (1)
 9:   HPOSI   EQU     SHOOTF+1        ; H.Position Data (1)
10:   VPOSI   EQU     HPOSI+1         ; V.Position Data (1)
11:   SWDATA  EQU     VPOSI+1         ; Switch Data (2)
12:
13:
14:
15:
16:
17: ;=====================================================;
18: ;                                                     ;
19: ;         ****** GUN SHOOT ADDRESS SEARCH *****       ;
20: ;                                                     ;
21: ;=====================================================;
22:
23: GUNS::
24:           LD      HL,HVCNT
25:           LD      DE,HVDATA
26:           LD      C,32
27: GUNS1::
28:           LD      A,(CDATA)       ; Color Data Flag Work
29:           DEC     A
30:           RET     NZ
31: GUNSPP:
32:           IN      A,(0DDH)
33:           AND     040H            ; bit 6, a Player 1 Pulse
34:           JP      NZ,GUNSPP       ; wait for low
35:           LD      A,(HL)
36:           CP      C
37:           JR      NC,GUNSPP
38:           INC     (HL)            ; Counter Up
39:           IN      A,(07FH)        ; H.Counter Read
40:           LD      (DE),A          ; RAM Save
41:           INC     DE
42:           IN      A,(07EH)        ; V.Counter Read
43:           LD      (DE),A          ; RAM Save
44:           INC     DE
45: GUNS2:
46:           IN      A,(0DDH)
47:           AND     040H            ; bit 6
48:           JP      Z,GUNS2
49:           JP      GUNSPP
50:
```

```
 51: ;=====================================================;
 52: ;               ***** V & H DATA CHECK *****           ;
 53: ;=====================================================;
 54: GUNCHK::
 55:           LD      A,(SHOOTF)      ; Gun Shoot Flag
 56:           OR      A
 57:           RET     Z
 58:           XOR     A
 59:           LD      (SHOOTF),A
 60: GUNCHKTI:
 61:           LD      A,(HVCNT)       ; V & H Counter
 62:           CP      5
 63:           RET     C
 64: ;
 65:           DEC     A
 66:           LD      B,A
 67:           LD      HL,HVDATA+1
 68:           LD      E,0H
 69: GUNNCH:
 70:           LD      A,(HL)          ; V.Counter
 71:           INC     E
 72:           LD      C,A
 73:           INC     HL
 74:           INC     HL
 75:           XOR     A
 76:           LD      A,(HL)
 77:           SUB     C
 78:           CP      3
 79:           CALL    NC,GCWKL2
 80:           DJNZ    GUNNCH
 81:           CALL    GCHKL2
 82: ;
 83:           CALL    CHGN
 84: ;
 85:           LD      A,(HL)          ; H-COUNTER READ
 86:           CP      0A0H
 87:           RET     NC
 88:           AND     A,A             ; CARRY FLAG = CLEAR
 89:           SUB     016H            ; A=A-14H
 90: ;
 91: ;SUB 016H-004H :Right ?? ?????
 92: ;SUB 016H+004H :Left ?? ?????
 93: ;
 94:           SLA     A               ; A=A*2
 95:           LD      B,A
 96:           REPT    5
 97:           RRA
 98:           ENDM
 99:           AND     A,07H           ; CARRY FLAG = CLEAR
100:           ADD     B               ; A=A+A/16
101:           LD      (HPOSI),A
102:           INC     HL
103:           LD      A,(HL)          ; V.Counter Read
104:           AND     A,A             ; CARRY FLAG = CLEAR
105:           ;SUB    018H            ; A=A-18H
106:           SUB     001H
107:           LD      (VPOSI),A
108:                                   ; * Color Data Set * ;
109:           XOR     A
110:           OUT     (0BFH),A
111:           LD      A,0C0H
112:           OUT     (0BFH),A
113:           LD      BC,020BEH
114:           LD      HL,COLORTBL     ; Color Data Table
115:           OTIR
116:           RET
117: COLORTBL:
118:           DEFB    000H,000H,000H,000H,000H,000H,000H,000H
119:           DEFB    000H,000H,000H,000H,000H,000H,000H,000H
120:           DEFB    000H,000H,000H,000H,000H,000H,000H,000H
121:           DEFB    000H,000H,000H,000H,000H,000H,000H,000H
122:
123:
124:
```

```
125:
126:
127: GCHKL2:
128:            PUSH    HL
129:            LD      HL,GCHKCT
130:            LD      A,(HL)
131:            CP      5
132:            JP      NC,GCHL1
133:            INC     (HL)
134:            LD      HL,GCHKWK
135:            PUSH    DE
136:            LD      D,0
137:            LD      E,A
138:            ADD     HL,DE
139:            POP     DE
140:            LD      (HL),E
141: GCHL1:
142:            POP     HL
143:            LD      E,0H
144:            RET
145: ;
146:
147: CHGGN:
148:            LD      A,(HL)
149:            JP      CHGN4
150: ;
151: CHGN:
152:            LD      HL,GCHKWK
153:            LD      C,0
154:            LD      A,(GCHKCT)
155:            CP      1
156:            JP      Z,CHGGN
157:            LD      B,A
158: CHGN1:
159:            LD      A,(HL)
160: CHGN2:
161:            INC     HL
162:            CP      (HL)
163:            JP      C,CHGN3
164:            DJNZ    CHGN2
165:            JP      CHGN4
166: CHGN3:
167:            INC     C
168:            DJNZ    CHGN1
169: CHGN4:
170:            RRCA
171:            AND     07FH
172:            LD      B,C
173:            LD      C,A
174:            LD      A,B
175:            OR      A
176:            JP      Z,CHGN6
177:            LD      HL,GCHKWK
178:            XOR     A
179: CHGN5:
180:            ADD     A,(HL)
181:            DJNZ    CHGN5
182:            ADD     A,C
183: CHGN7:
184:            ADD     A,A
185:            LD      C,A
186:            LD      B,0
187:            LD      HL,HVDATA
188:            ADD     HL,BC
189:            RET
190: CHGN6:
191:            LD A,C
192:            JP CHGN7
193:
194:
195:
196:
```

```
197: ;=======================================================;
198: ;                                                       ;
199: ;               ***** GUN CHECK *****                   ;
200: ;                                                       ;
201: ;=======================================================;
202: GUNINT::
203:           LD      HL,CDATA        ; Flash States Work
204:           LD      A,(HL)
205:           LD      (HL),2          ;
206:           DEC     A               ; CP 1
207:           RET     Z               ;(CDATA)=1 ---> (CDATA)=2 , RET
208:           LD      (HL),0          ;
209:           DEC     A               ; CP 2
210:           RET     Z               ;(CDATA)=2 ---> (CDATA)=0 , RET
211:                                   ;
212:           CALL    SWSET
213:           AND     010H            ; bit 4
214:           RET     Z               ;(CDATA)=0 , TRI.=OFF ---> RET
215:                                   ;
216:           XOR     A
217:           LD      (HVCNT),A
218:           LD      A,1             ; A.res = 1
219:           LD      (CDATA),A       ;(CDATA)=0 , TRI.=ON ---> (CDATA)=1 , RET
220:           LD      (SHOOTF),A      ; Gun Shoot Flag Set
221: ; * Screen Flash *
222:           XOR     A
223:           OUT     (0BFH),A
224:           LD      A,0C0H
225:           OUT     (0BFH),A
226:           LD      A,03FH
227:           OUT     (0BEH),A
228: ; * RAM Clear *
229:           LD      HL,HVCNT
230:           LD      DE,HVCNT+1
231:           LD      BC,001H+040H+001H+10
232:           LD      (HL),0
233:           LDIR
234:           RET
235:
236:
237:
238:
```

```
;=======================================================;
;                                                       ;
;                                                       ;
;                                                       ;
;=======================================================;
;                                                       ;
;               1.0.1.0         now                     ;
;               ------- CPL                            ;
;               0.1.0.1         now                     ;
;               1.1.0.0         old                     ;
;               ------- AND                            ;
;               0.1.0.0                                 ;
;               ------- CPL                            ;
;               1.0.1.1                                 ;
;=======================================================;
SWSET:
          IN      A,(0DCH)        ;
          AND     010H            ; bit 4
          LD      HL,SWDATA       ; Switch data save area
          CPL
          LD      C,A             ; Data save
          XOR     (HL)
          LD      (HL),C          ; New SW.data save
          INC     HL
          AND     C               ; ACC : '0' --> '1' change but data
          LD      (HL),A          ; Data save
          RET
;=======================================================;
;             *** Interrupt Jump Check ***              ;
;=======================================================;
:ORG      038H
          PUSH    AF
          LD      A,(CDATA)       ; Gun Set ?
          DEC     A
          JR      NZ,INSS
          EX      (SP),HL         ; Stack Pointer Change
          POP     HL              ;
          LD      HL,GUNS1        ;
          EX      (SP),HL         ;
          PUSH    AF
INSS:
          POP     AF
          JP      INT38
```

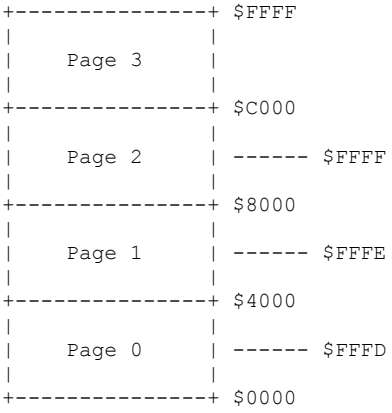
ANNEXE D – EXTRAIT DE CODE DE LECTURE DE LA TRACK BALL

```
 1:
 2: ;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~;
 3: ; TRACK BALL SWITCH READ                                                 ;
 4: ;         RET     P1_TRX ~ PLAYER 1 CONDITION (-7FH ~ 7FH)               ;
 5: ;                 P1_TRY ~ PLAYER 1 CONDITION (-7FH ~ 7FH)               ;
 6: ;                 P2_TRX ~ PLAYER 2 CONDITION (-7FH ~ 7FH)               ;
 7: ;                 P2_TRY ~ PLAYER 2 CONDITION (-7FH ~ 7FH)               ;
 8: ;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~;
 9: 01D0'             T_BALL::
10: 01D0'             TRACK_X1:
11: 01D0' 3E 0D               LD      A,00001101B
12: 01D2' D3 3F               OUT     (PSWC),A
13: 01D4' 06 19               LD      B,25
14: 01D6' 10 FE               DJNZ    $
15: 01D8' DB DC               IN      A,(P1_SWPT)
16:
17:
18:
19: 01DA' E6 0F               AND     0FH
20:                           REPT    4
21:                           RRCA
22:                           ENDM
23: 01DC' 0F          *       RRCA
24: 01DD' 0F          *       RRCA
25: 01DE' 0F          *       RRCA
26: 01DF' 0F          *       RRCA
27: 01E0' 57                  LD      D,A
28: 01E1' 3E 2D               LD      A,00101101B
29: 01E3' D3 3F               OUT     (PSWC),A
30: 01E5' 06 0D               LD      B,13
31: 01E7' 10 FE               DJNZ    $
32: 01E9' DB DC               IN      A,(P1_SWPT)
33: 01EB' E6 0F               AND     0FH
34: 01ED' B2                  OR      D
35: 01EE' ED 44               NEG
36: 01F0' 32 C018             LD      (P1_TRX),A
37: 01F3'             TRACK_Y1:
38: 01F3' 3E 0D               LD      A,00001101B
39: 01F5' D3 3F               OUT     (PSWC),A
40: 01F7' 06 0D               LD      B,13
41: 01F9' 10 FE               DJNZ    $
42: 01FB' DB DC               IN      A,(P1_SWPT)
43: 01FD' E6 0F               AND     0FH
44:                           REPT 4
45:                           RRCA
46:                           ENDM
47: 01FF' 0F          *       RRCA
48: 0200' 0F          *       RRCA
49: 0201' 0F          *       RRCA
50: 0202' 0F          *       RRCA
51: 0203' 57                  LD      D,A
52: 0204' 3E 2D               LD      A,00101101B
53: 0206' D3 3F               OUT     (PSWC),A
54: 0208' 06 0D               LD      B,13
55: 020A' 10 FE               DJNZ    $
56: 020C' DB DC               IN      A,(P1_SWPT)
57: 020E' E6 0F               AND     0FH
58: 0210' B2                  OR      D
59: 0211' ED 44               NEG
60: 0213' 32 C019             LD      (P1_TRY),A
61: 0216'             TRACK_X2:
62: 0216' 3E 07               LD      A,00000111B
63: 0218' D3 3F               OUT     (PSWC),A
64: 021A' 06 19               LD      B,25
65: 021C' 10 FE               DJNZ    $
66: 021E' DB DC               IN      A,(P1_SWPT)
67: 0220' E6 C0               AND     11000000B
68: 0222' 0F                  RRCA
69: 0223' 0F                  RRCA
70: 0224' 5F                  LD      E,A
71: 0225' DB DD               IN      A,(P2_SWPT)
72: 0227' E6 03               AND     11B
73: 0229' 0F                  RRCA
```

```
 74: 022A' 0F                RRCA
 75:
 76:
 77:
 78: 022B' 83                OR      E
 79: 022C' 57                LD      D,A
 80: 022D' 3E 87             LD      A,10000111B
 81: 022F' D3 3F             OUT     (PSWC),A
 82: 0231' 06 0D             LD      B,13
 83: 0233' 10 FE             DJNZ    $
 84: 0235' DB DC             IN      A,(P1_SWPT)
 85: 0237' E6 C0             AND     11000000B
 86: 0239' 5F                LD      E,A
 87: 023A' DB DD             IN      A,(P2_SWPT)
 88: 023C' E6 03             AND     11B
 89: 023E' B3                OR      E
 90: 023F' 07                RLCA
 91: 0240' 07                RLCA
 92: 0241' B2                OR      D
 93: 0242' ED 44             NEG
 94: 0244' 32 C01A           LD      (P2_TRX),A
 95: 0247'             TRACK_Y2:
 96: 0247' 3E 07             LD      A,00000111B
 97: 0249' D3 3F             OUT     (PSWC),A
 98: 024B' 06 19             LD      B,25
 99: 024D' 10 FE             DJNZ    $
100: 024F' DB DC             IN      A,(P1_SWPT)
101: 0251' E6 C0             AND     11000000B
102: 0253' 0F                RRCA
103: 0254' 0F                RRCA
104: 0255' 5F                LD      E,A
105: 0256' DB DD             IN      A,(P2_SWPT)
106: 0258' E6 03             AND     11B
107: 025A' 0F                RRCA
108: 025B' 0F                RRCA
109: 025C' B3                OR      E
110: 025D' 57                LD      D,A
111: 025E' 3E 87             LD      A,10000111B
112: 0260' D3 3F             OUT     (PSWC),A
113: 0262' 06 0D             LD      B,13
114: 0264' 10 FE             DJNZ    $
115: 0266' DB DC             IN      A,(P1_SWPT)
116: 0268' E6 C0             AND     11000000B
117: 026A' 5E                LD      E,A
118: 026B' DB DD             IN      A,(P2_SWPT)
119: 026D' E6 03             AND     11B
120: 026F' B3                OR      E
121: 0270' 07                RLCA
122: 0271' 07                RLCA
123: 0272' B2                OR      D
124: 0273' ED 44             NEG
125: 0275' 32 C01B           LD      (P2_TRY),A
126: 0278' C9                RET
127:
```

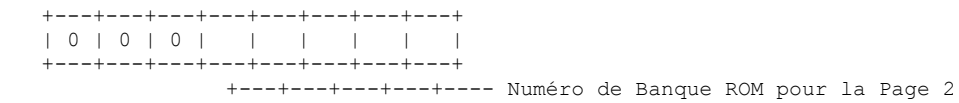
NOTES POUR LE DEVELOPPEUR

(1) Registres de Contrôle Mémoire

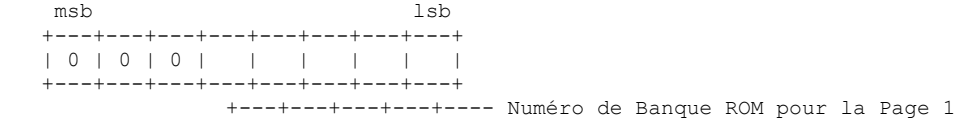


(2) Registres de Contrôle Mémoire

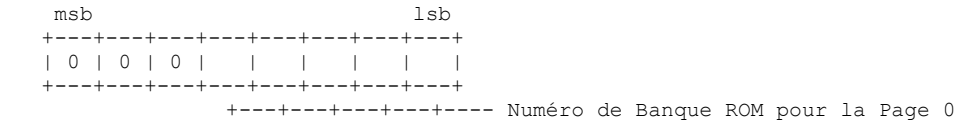
* $FFFF – Registre de Contrôle de la Page 2



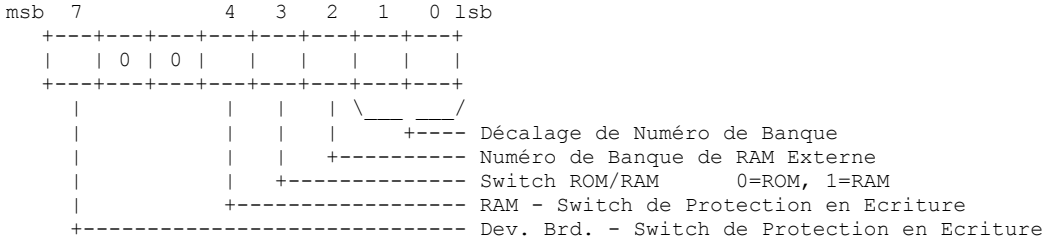
* $FFFE – Registre de Contrôle de la Page 1



* $FFFD – Registre de Contrôle de la Page 0



* $FFFC – Registre de Contrôle de la Page 2 de la RAM



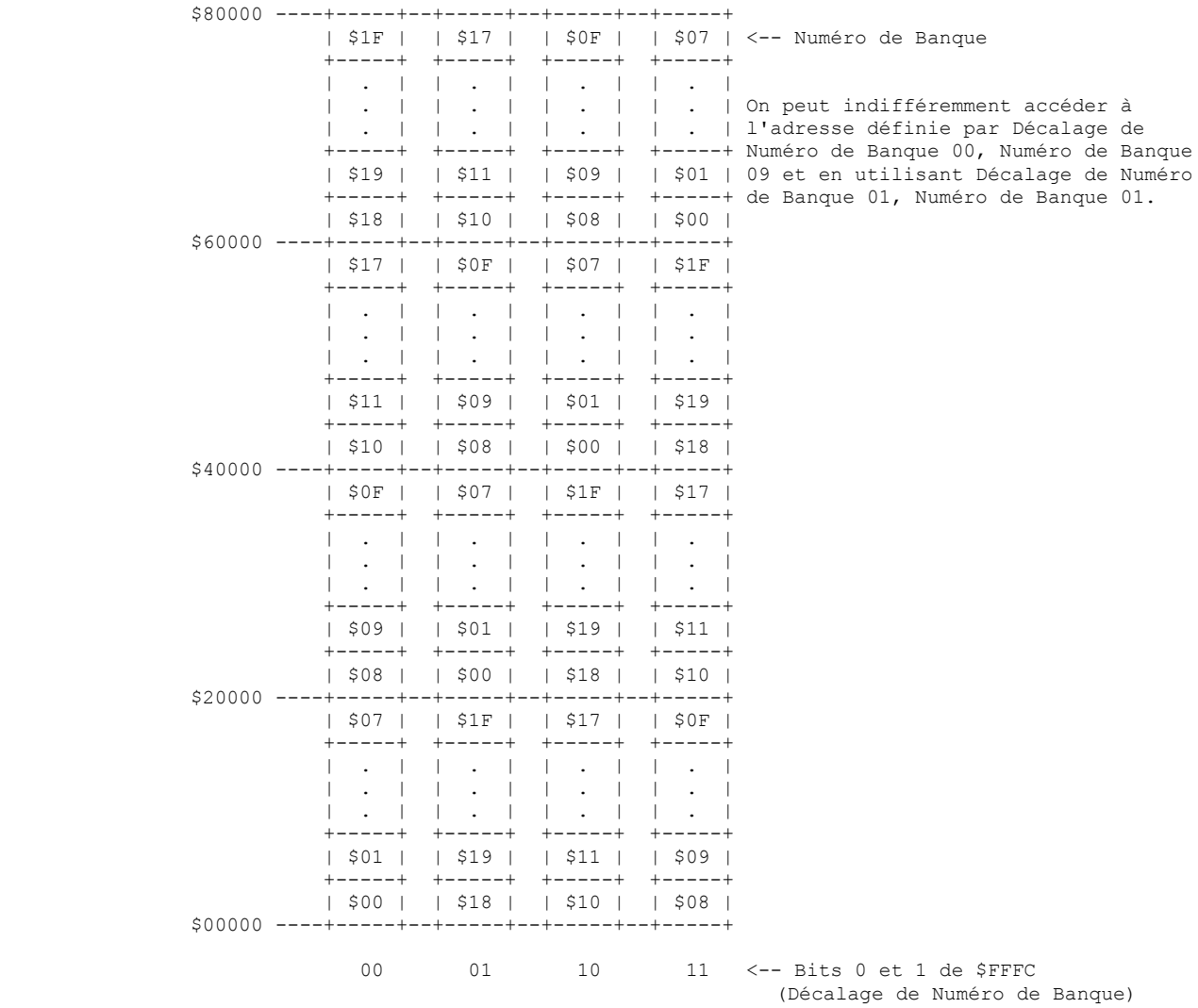
NOTE: Quand le système est mis sous tension, le programme en ROM système configure les données suivantes:

$FFFF = 2 $FFFE = 1 $FFFD = 0 $FFFC = 0

Le bit 7 de $FFFC est le bit de protection en écriture de la *ROM Board* (Uniquement pour la *development board-carte de développement*)

0 = Lecture/Ecriture 1 = Lecture seule

(3) La ROM de 4M Bit



(4) Adresses $0000 à $03FF du Z80

Le premier Kilooctet (1024 octets) du Z80A est toujours mappé au premier Kilooctet de ROM pour garantir que les vecteurs de reset et d'interruption soient toujours disponibles.

Par exemple, si la Banque n°8 (Adresses ROM $20000 à $24000) était assignée à la Page 0 (Adresses $0000 à $4000 du Z80), on ne pourrait pas accéder aux adresses ROM $20000 à $203FF car les adresses $0000 à $03FF du Z80 resteraient mappées aux adresses ROM $00000 à $003FF.