Daniel Martin

LE LIVRE DU MSX 2

BCM s.c.

INTRODUCTION

Ce livre n'est pas une suite au livre du MSX, c'est une toute nouvelle mouture, spécialement destinée aux possesseurs d'un système MSX2.

Comme dans le livre du MSX, nous avons réservé une grande place à la description des coprocesseurs et plus spécialement du gestionnaire vidéo VDP 9938 qui à lui seul nécessiterait un livre entier.

Par contre, nous avons réduit la partie réservée au Basic. En effet, tout ce qui a été décrit dans le premier livre reste valable en MSX2. Nous nous sommes donc contentés de vous fournir quelques compléments d'information qui s'ajoutent à ceux de l'autre ouvrage.

En échange, nous avons développé de façon intensive la description des vecteurs et des adresses internes du système. Ce livre constitue donc un répertoire exhaustif des points d'entrées du logiciel interne du MSX2.

Nous avons réservé un chapitre complet aux mécanismes de gestion de la mémoire centrale.

En espérant que vous trouverez dans cet ouvrage la réponse à la plupart de vos questions, nous vous souhaitons une excellente lecture.

****

Remarque : Ce livre ne contient volontairement aucune spécification sur le système disque. Ces informations valables en MSX1 comme en MSX2, figurent dans un autre ouvrage de la collection : « Le livre du disque MSX » de Manu Devos.

Les informations sur le Basic et les trucs et astuces du livre du MSX1 ne sont pas repris dans cet ouvrage. Reportez-vous au livre 1 pour de plus amples informations.

****

CHAPITRE 1

STRUCTURE INTERNE DES SYSTÈMES MSX2

1 Généralités

Le système MSX2, comme son ancêtre le MSX1, est architecturé autour d'un processeur ZILOG Z80. Ce processeur 8 bits âgé aujourd'hui de plus d'une dizaine d'années peut paraître désuet et faible en comparaison des processeurs 16 bits qui équipent les ATARI ST et autres MACINTOSH.

Il faut cependant garder plusieurs points à l'esprit :

- Le système MSX est un ordinateur personnel et n'a pas en principe besoin d'une puissance de calcul démesurée.
- Dans les ordinateurs cités plus haut, une grande partie des ressources du processeur est utilisée pour maintenir le système graphique (GEM, MAC, WINDOWS...) alors que le MSX utilise un système de communication classique et peu gourmand (MSX-DOS et Basic).
- Le système MSX dispose d'un coprocesseur vidéo très sophistiqué alors que le 68000 d'un ATARI ST par exemple doit gérer seul toute sa mémoire d'écran.

Le processeur Z80 équipé d'un bus de données de 8 bits et d'un bus d'adresse de 16 bits peut en principe adresser 64 Ko de mémoire. Cependant, la conception géniale du matériel entourant le processeur permet de porter cet espace à 1 Mo (16 * 64 Ko) et même dans certains systèmes à 4 Mo. La technique utilisée pour accomplir ce miracle est exposée clairement dans le chapitre 5.

Autour du processeur Z80, on trouve le bus d'adresse, de la mémoire vive (RAM), de la mémoire morte (ROM) contenant le Basic et le Bios (primitives de base du système).

Le Z80 possède aussi une série (256) de ports de communication. Ces ports servent principalement à interfacer le processeur central avec les coprocesseurs et les périphériques divers qui peuvent équiper un système MSX.

Les coprocesseurs que l'on trouve dans tous les systèmes MSX2 sont :

- Le PSG (Programmable Sound Generator) qui s'occupe de la gestion sonore du système et qui est décrit en détail au chapitre 2.
- Le VDP (Video Display Processor) qui s'occupe de la gestion de l'image (texte et graphique) et dont la description fait l'objet du chapitre 3.

A côté de ces processeurs, on trouve quelques circuits moins importants qui sont décrits au chapitre 4 ainsi que l'interface disque (qui équipe en standard presque tous les systèmes MSX2 et qui fait l'objet d'un ouvrage séparé).

La structure des ports d'entrée/sortie et de la mémoire est décrite sous forme de table dans la suite de ce chapitre.

Le système MSX est conçu pour être évolutif. Cette décision oblige les concepteurs à respecter un certain nombre de préceptes dans la réalisation du logiciel interne du système (FIRMWARE). La description de ces principes fait l'objet du chapitre 5.

Le chapitre 6 concerne plus particulièrement la gestion de la mémoire centrale et des slots.

Le chapitre 7 contient les conseils donnés par Microsoft aux sociétés de développement de logiciels ainsi que quelques compléments sur la structure du Basic qui ne figuraient pas dans le premier livre.

Ce dernier chapitre vous apporte également une série de conseils et de trucs pour mieux tirer parti de votre MSX.

2 Table des adresses des ports d'entrée/sortie

Les adresses dont le nom est accompagné du symbole # ne sont pas disponibles en MSX1. Ces adresses peuvent être utilisées pour ajouter des périphériques si le système les décode complètement.

Non défini	00-3F	
Réservé	40-7F	
RS-232C	80	8251 DATA
	81	8251 ETAT/COMMANDE
	82	Etat de lecture, masque d'interruption
	83	Non défini
	84	8253

	85	8253
	86	8253
	87	8253
VDP (9918) adaptateur MSX1	88-8B	REM : Adresses de l'adaptateur vidéo MSX1. Pour garder la compatibilité entre le vrai système MSX2 et le système MSX1 avec l'adaptateur MSX2 (qui a son chip vidéo à une adresse supérieure), tous les logiciels MSX2 accèdent au chip vidéo par l'adresse chargée aux adresses 6 et 7 de la ROM.
Modem	8C-8D	
Réservé	8E-8F	
Port imprimante	90	(w) bit 0 : Strobe
		(r) bit 1 : Etat
	91	imprime les datas
Réservé #	92-97	
VDP (9938)	98-9B	Voir chapitre 3
Réservé #	9C-9F	
Générateur de son	A0	Adresse latch
(AY-3-8910)	A1	Ecriture des datas
	A2	Lecture des datas
	A3	
Réservé #	A4-A7	
Port parallèle	A8	Port A
(8255)	A9	Port B
	AA	Port C
	AB	Mode
MSX-ENGINE #	AC-AF	REM : pour les systèmes sans MSX-ENGINE, cette zone peut être une image du 8255.

Mémoire externe	B0	Port A : adresses A0-A7
(SONY DATAPACK)	B1	Port B : adresses A8-A10 A13-A15 Contrôle & R/W
	B2	Port C : Adresses A11-A12 Données D0-D7
	B3	Mode
Horloge calendrier	B4	Adresse latch
	B5	Données
Réservé	B6-B7	
Contrôle du crayon optique	B8	Lecture/Ecriture
	B9	Lecture/Ecriture
	BA	Lecture/Ecriture
	BB	Lecture/Ecriture
Videodisk JVC	BC	Port A
(8255)	BD	Port B
	BE	Port C
	BF	Mode
MSX Audio	C0-C1	
Réservé	C2-C7	
Interface MSX	C8-CF	
Contrôleur de disques	D0-D7	REM : Ces adresses sont réservées aux contrôleurs de disques. Pour tout complément d'information sur le système disque, reportez-vous au livre du disque MSX de Manu Devos.
Réservé	D8-DB	générateur de caractères japonais.

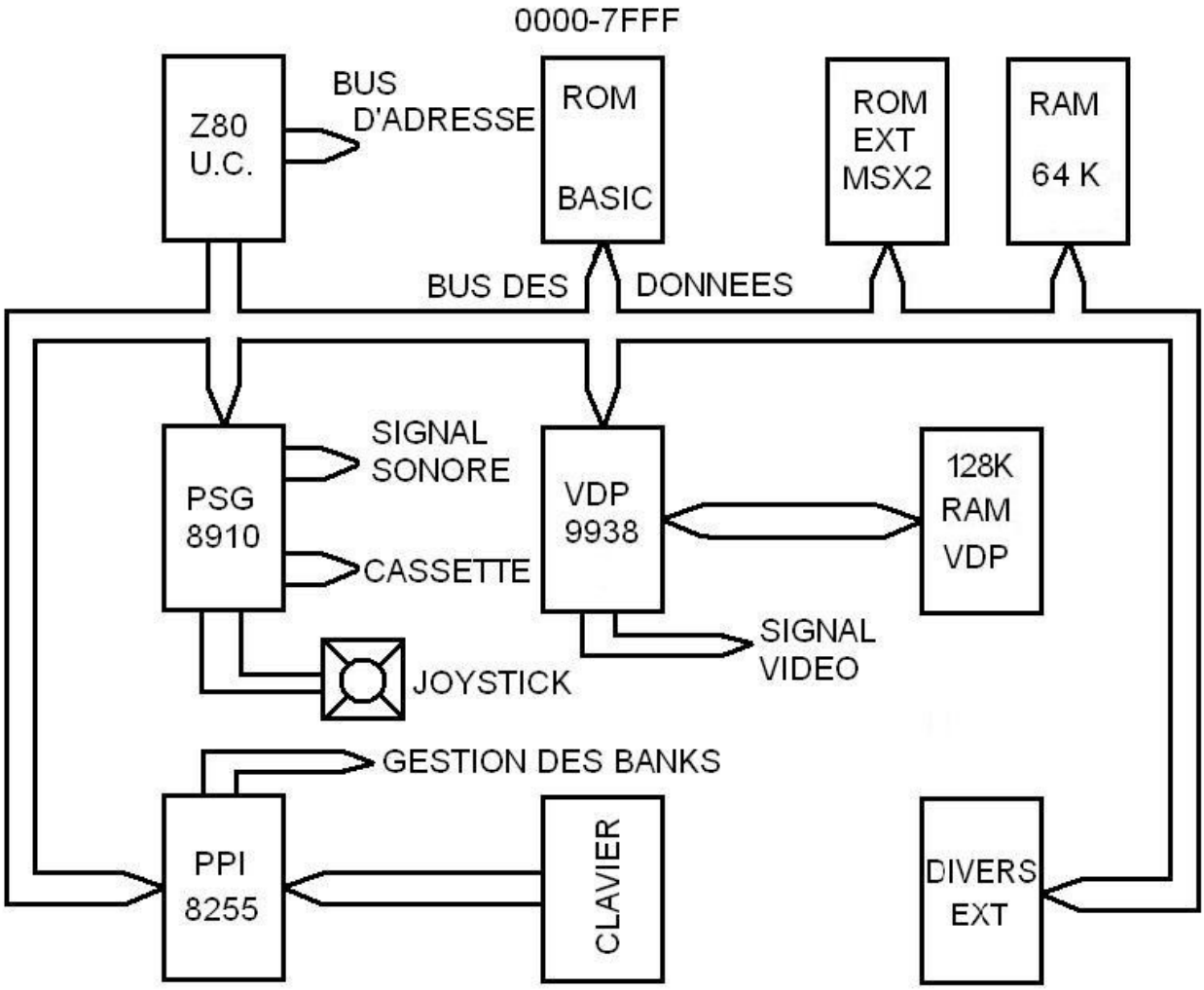
Réservé	DC-F4	
Contrôle système	F5	Dispositif interne uniquement pour l'écriture. Ecrire 1 pour le rendre accessible. b0 ROM Kanji b1 réservé pour Kanji b2 MSX-Audio b3 b4 Interface MSX b5 RS232C b6 Crayon optique lumineux b7 Horloge/Calendrier. Depuis que l'horloge/calendrier fait partie du standard MSX2, ce bit n'est plus nécessaire. Il est cependant utile en MSX1. REM : Le but de ces bits est de prévenir un conflit entre les périphériques d'entrée/sortie qui pourraient être installés de façon interne et ceux installés par l'intermédiaire des ports cartouches. Les premiers peuvent être déconnectés par ces bits.
Bus d'entrée/sortie couleurs	F6	
Contrôle audio/vidéo	F7	b0 audio droite → L mixing on (écriture) b1 audio gauche → L mixing on (écriture) b2 sélection d'entrée vidéo → L 21p RGB (écriture) b3 sélection du sens vidéo → L Pas d'entrée (lecture) b4 contrôle audio-vidéo → L TV (écriture) b5 contrôle Ym → L TV (écriture) b6 contrôle Ys → L Super (écriture) b7 sélection vidéo → L TV (écriture)
Réservé	F8-FB	
Carte mémoire	FC-FF	

3 Organisation de la RAM MSX

FFFF	Réservé pour les slots secondaires les registres de sélection (4 octets)	
FFFC		
FFF8	Réservé pour des extensions futures (4 octets)	
FFF7	Adresse du slot de la ROM principale	
FFE7	VDP - MSX2 - 9938 registres VDP 8-23 (16 octets)	
FFD9	Programme XXXINT (14 octets)	
FFD4	Vecteur crochet ENAINT (5 octets)	
FFCF	Vecteur crochet DISINT (5 octets)	
FFCA	Entrée de l'appel du Bios étendu (5 octets)	
FD9A	Zone le la RAM Vecteurs-crochets (560 octets) 112 vecteurs-crochets	HOKJMP
F380	Zone de travail système (2856 octets)	INIDAT
8000	Zone utilisateur	
0000	Zone du RAM Disk	

4 Organisation de la ROM MSX

8000	
7FFF	Instruction pour le saut (Jump)
7FFD	vers l'entrée du Bios
4000	ROM BASIC (interpréteur)
3FFF	Contenu principal du Bios
01B6	
	trou de 66 octets
0171	BIOENT.MAC - Ajout d'entrées au Bios (MSX2)
015C	
0000	BIOENT.MAC - Vecteurs de saut au Bios



CHAPITRE 2

LE GÉNÉRATEUR SONORE AY3-8910

1 Généralités

Le générateur sonore AY3-8910 de General Instrument, en abrégé PSG (Programmable Sound Generator), est un circuit LSI (circuit à haute intégration) qui peut produire une grande variété de sons complexes sous le contrôle d'un programme approprié.

Le PSG est un composant relativement facile à interfacer avec n'importe quel système à microprocesseur. Sa flexibilité est telle qu'il est souvent utilisé dans les applications les plus diverses.

La sortie sonore analogique (c'est-à-dire un signal non-digital) est effectuée par l'intermédiaire d'un convertisseur DAC (Digital Analogic Converter) sur quatre bits : celui-ci permet une grande variété d'effets sonores.

Une des principales caractéristiques du circuit est qu'une fois ses commandes de génération reçues, il gère lui-même les effets sonores, laissant le processeur libre pour continuer d'autres tâches. Ainsi, le PSG peut produire des sons relativement longs en n'affectant en rien la vitesse d'exécution du programme.

Le PSG possède trois voies mixables et permet la sortie de trois sons simultanés, donc la création d'un accord musical simple, majeur ou mineur.

Le PSG est un coprocesseur dont la gestion se fait au moyen de divers registres programmables. Ces registres sont au nombre de 16 et seront décrits en détail au long de ce chapitre.

2 La théorie du son

Pour bien comprendre la programmation et l'utilisation du générateur sonore, il est indispensable d'analyser dans les grandes lignes tous les paramètres qui définissent un phénomène sonore ou, plus simplement, une note de musique.

2.1 Qualités du son

Un son consiste en la propagation, à partir d'une source (en ce qui nous concerne, la membrane du haut-parleur), d'ondes matérielles périodiques longitudinales avec une vitesse dont la grandeur dépend du milieu de propagation.

Exemple : vibrations d'un ressort après avoir exercé sur lui une compression ou une traction.

On entend par qualité du son, les diverses caractéristiques par lesquelles différents sons se distinguent les uns des autres.

Elles sont au nombre de trois : la HAUTEUR, le VOLUME et le TIMBRE.

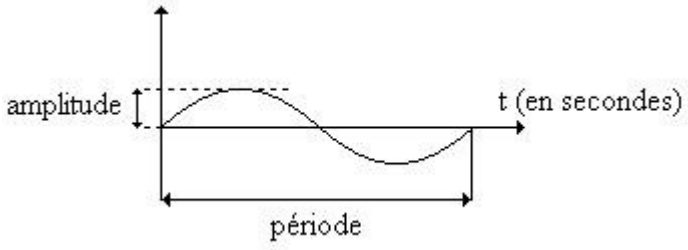
2.1.1 La hauteur

Lorsque nous frappons successivement quelques touches d'un piano, nous percevons des sons différents les uns des autres. Les sons se distinguent par leur hauteur ou leur fréquence.

Les sons graves correspondent aux basses fréquences et les sons aigus aux fréquences élevées.

Dans une note de musique, la hauteur est un facteur essentiel.

Une note musicale peut être décrite comme une oscillation qui se caractérise par une fréquence, une période et une amplitude.



La période est le temps, exprimé en secondes, d'une oscillation complète.

L'amplitude dépend du volume et n'entre pas dans la définition de la hauteur d'une note.

La fréquence et la période sont liées entre elles par le rapport suivant :

FREQUENCE = 1 / PERIODE

Notons que l'impression musicale résultant de l'audition de deux sons de hauteur différentes dépend, non pas des valeurs absolues de celles-ci, mais bien de leur rapport. C'est ce rapport qui l'on appelle INTERVALLE DE DEUX TONS.

Lorsque l'un des sons présente une fréquence double de celle de l'autre, l'intervalle est appelé OCTAVE.

La gamme, dite naturelle, est formée de différents sons présentant les intervalles suivants par rapport au premier son :

DO	RE	MI	FA	SOL	LA	SI	DO
1	9/8	5/4	4/3	3/2	5/3	15/8	2

Les octaves se suivent en présentant les mêmes intervalles. Les différentes octaves sont affectées respectivement des indices 1, 2, 3... ou des indices -1, -2, -3... suivant qu'ils sont supérieurs ou inférieurs à l'octave 0, octave de référence.

2.1.2 Le volume

Une même touche de piano produira un effet sonore différent suivant qu'elle est frappée légèrement ou fortement.

Dans le premier cas, le son est faible, dans le second, il est fort. Les deux sons diffèrent par leur VOLUME, c'est-à-dire leur intensité.

Le volume détermine donc la grandeur de l'effet sonore.

2.1.3 Le timbre

L'oreille humaine distingue des sons de même hauteur et de même intensité mais émis par des instruments différents. Le LA du violon est perçu différemment du LA du piano.

Cette différence est due au fait qu'une note émise par un instrument n'est jamais pure (ou simple) mais est composée d'une multitude de sons simples, appelés HARMONIQUES, superposés au son pur initial appelé FONDAMENTAL.

Le timbre d'un son dépend des fréquences et des intensités relatives des harmoniques qui accompagnent le son fondamental.

2.2 Les bruits

On distingue les bruits des sons musicaux par le fait que les premiers ne sont pas périodiques, c'est-à-dire qu'ils ne se reproduisent pas exactement à intervalles réguliers dans le temps.

La distinction entre une note et un bruit n'est pas toujours très nette. Bien que l'oreille humaine soit sensible à tous les sons dont la fréquence est comprise entre 20 et 20 000 Hertz, l'effet musical n'est plus perçu pour les sons périodiques dont la fréquence est trop basse ou trop élevée. De plus, les limites varient d'un auditeur à l'autre.

2.3 Durée et attaque de la note

La durée d'un son présente une importance non négligeable quant à l'interprétation que l'oreille humaine pourra en faire.

Pour l'oreille, la durée d'un son dépendra de l'état physiologique de l'individu et des durées relatives des sons qui ont précédé le son considéré.

De plus, il faut noter que la perception sonore dépend également de l'attaque ou de la chute d'un son.

On entend par attaque (chute) la vitesse à laquelle le son considéré atteint un volume déterminé.

3 Structure interne du PSG

Le PSG est composé des éléments suivants :

a) GENERATEURS SONORES : au nombre de trois, ils sont respectivement appelés canal A, canal B et canal C. Ils produisent un signal carré dont la fréquence est programmable.b)

b) LE GENERATEUR DE BRUIT : il produit une modulation de fréquence aléatoire.

c) LE MELANGEUR (MIXER) : il permet de combiner les sorties des 3 canaux A, B et C du générateur sonore avec le générateur de bruit.

d) LE CONTROLEUR D'AMPLITUDE : il fournit au D/A (Digital Analog converter) la possibilité de contrôler l'amplitude par un modèle fixe ou variable. Le modèle d'amplitude fixe est contrôlé par le microprocesseur lui-même tandis que le modèle d'amplitude variable est obtenu par l'utilisation du générateur d'enveloppe.

e) LE GENERATEUR D'ENVELOPPE : il produit un modèle de variation d'amplitude appelé enveloppe qui peut être utilisé pour moduler la sortie de chaque mixer.

f) LES CONVERTISSEURS DIGITAUX/ANALOGIQUES : D/A : les 3 convertisseurs D/A produisent un signal de sortie sur 16 niveaux déterminés par le contrôleur d'amplitude.

g) LES PORTS D'ENTREE/SORTIE : le générateur sonore AY3-8912 possède un port d'entrée/sortie. Ce port ne joue aucun rôle dans la production sonore. Il sera analysé en même temps que le PPI (Programmable Peripheral Interface).

4 Les divers registres du PSG

Les registres qui permettent la programmation du PSG sont au nombre de 16 numérotés de R0 à R15.

Les registres R14 et R15 servent à la gestion des ports d'entrée/sortie et seront analysés par la suite.

Rem: Pour AY3-8912, il n'y a qu'un seul port de sortie. Le registre R15 n'est pas utilisé.

Pour produire un son, une combinaison des registres R0 à R15 doit être introduite dans le PSG. Chaque son doit être analysé de façon à séparer les différents paramètres qui le définissent. C'est-à-dire : la composante de bruit, de son, la fréquence, la forme et la durée des enveloppes.

Une fois cette analyse terminée, les registres peuvent être chargés et le son produit. Le schéma bloc figurant ci-après montre les interactions entre les différentes sections du PSG.

4.1 Les registres R0 à R5

Les registres R0 à R5 définissent la fréquence du son émis. Ils sont divisés en 3 paires : R0-R1 pour la canal A, R2-R3 pour le canal B et R4-R5 pour le canal C.

Les registres R0, R2 et R4 sont les registres de réglage fin de la fréquence et les 8 bits sont utilisés.

Les registres R1, R3 et R5 quant à eux sont les registres de réglage grossier (seuls les 4 bits de poids faible LSB sont utilisés).

Les valeurs chargées dans les registres R1, R3 et R5 sont donc comprises entre 0 et 15 (00 et FF en hexadécimal).

Dans le PSG, la fréquence d'un son est déterminée en divisant premièrement la fréquence de l'horloge interne par 16 et puis par F qui est la fréquence à programmer.

On applique la formule suivante :

PT = 3 579 545 / (16*F) où PT est la période de ton.

La valeur résultante des paires de registres étant codée sur 12 bits, PT doit être arrondi à l'unité avant d'être exprimé sur 12 bits au moyen de la fonction :

PT = BIN$(F, 12) = XXXX XXXX XXXX

\-------/ \-----------------/

RH RL

Les 8 bits de droite sont transmis dans les registres R0, R2 ou R4 et les 4 bits de gauche dans les registres R1, R3 ou R5.

ajustement grossier - canal - ajustement fin

B7 B6 B5 B4 B3 B2 B1 B0 B7 B6 B5 B4 B3 B2 B1 B0

\---------------/ non utilisé

PT11 PT10 PT9 PT8 PT7 PT6 PT5 PT4 PT3 PT2 PT1 PT0

12 bits pour le générateur sonore

Une autre façon de procéder pour charger les paires de registres consiste à calculer RL et RH comme suit :

RL = PT - (INT(PT/256)*256) et RH = INT(PT/256) ou encore RH = PT\256 (c'est bien le signe \ et NON / ! ! !).

Il suffit alors de transmettre RL dans R, R2 ou R4 et RH dans R1, R3 ou R5.

Exemple : Soit F = 440 Hz (Le « LA » international)

PT = 3 579 545 / (16*440)

PT = 3 579 545 / 7040

PT = 508.45

On arrondit : PT = 508

On calcule : RL = MOD (508, 256)

RL = 252

RH = 508 \ 256

RH = 1

Si c'est le canal A qui dot être programmé, R0 sera égal à 252 et R1 à 1.

Détermination de F minimum et de F maximum

Comme PT (Période de Ton) est exprimé sur 12 bits, la valeur que l'on peut charger ne peut excéder 4095, et la valeur minimale est égale à 1.

1 donne F max et 4095 donne F min.

1 = 3579545/(16*Fmax) → Fmax = 3579545/16*1 = 223421 Hz

4095 = 3579545/(16*Fmin) → Fmin = 3579545/16*4095 = 54,6 Hz

Il est évident qu'une fréquence de la valeur de Fmax est, comme nous l'avons vu dans le paragraphe traitant de la théorie du son, imperceptible par l'oreille humaine.

De plus, la bande passante des petits amplis audio ou télévisuels dépasse rarement les 5000 Hz. Dès

lors, nous nous fixerons comme valeur maximum de fréquence cette valeur de 5000 Hz.

Un simple calcul donne PT = 44.

Donc les valeurs de PT seront comprises entre 44 et 4095.

4.2 Le registre R6

Le registre R6 du PSG permet de programmer une fréquence de bruit de la façon suivante :

Seuls les 5 bits de poids faible du registre R6 sont utilisés pour programmer le générateur de bruit.

Registre de période de bruit	
B7 B6 B5	B4 B3 B2 B1 B0
non utilisé	5 bits pour le générateur de bruit

La formule appliquée pour trouver la période de bruit est semblable à celle utilisée pour calculer la période de Ton (PT) :

Soit PB la période de bruit :

PB = 3 579 545 / (16*Fb) où Fb est la fréquence de bruit désirée.

Comme seuls les 5 premiers bits de R6 sont utilisés pour déterminer PB, les valeurs de PBmax et PBmin seront comprises entre 1 et 2^5 -1 soit 31.

Par la formule on détermine Fbmax et Fbmin :

Fbmax = 3579545 / (16 * PBmin) = 223721 Hz

Fbmin = 3579545 /(16 * PBmax) = 7216 Hz

Il est à noter que les restrictions applicables pour Fmax sont aussi à observer pour Fbmax (Fréquence de bruit maximum).

4.3 Le registre R7

Le registre R7 permet de contrôler la fonction de mixage entre les 3 générateurs sonores et les 3 générateurs de bruit. Le mixage, comme décrit plus haut, autorise la combinaison de chaque canal A, B ou C avec un générateur de bruits. Le registre R7 contrôle également les ports d'entrée/sortie A et B dont nous parlerons par la suite.

Voici un tableau résumant les effets du registre R7 :

BIT = 0		=1
7	Port B entrée	Port B sortie
6	Port A entrée	Port A sortie
5	Bruit sur canal C ON	Bruit sur canal C OFF
4	Bruit sur canal B ON	Bruit sur canal B OFF
3	Bruit sur canal C ON	Bruit sur canal C OFF
2	Son sur canal C ON	Son sur canal C OFF
1	Son sur canal B ON	Son sur canal B OFF
0	Son sur canal C ON	Son sur canal C OFF

Remarque : Mettre un canal sur OFF ne suffit pas pour arrêter l'émission de celui-ci. Il faut écrire 0 dans le registre de contrôle d'amplitude.

Exemple : Je désire produire sur le canal A du son et pas de bruit; sur le canal B du bruit et pas de son; sur le canal C du bruit et du son.

Je dois donc charger le registre R7 avec la valeur suivante :

Val. déc.	128	64	32	16	8	4	2	1
Bit	B7	B6	B5	B4	B3	B2	B1	B0
	X	X	0	0	1	0	1	0

(X) = sans importance

R7 = 10, en base 10 ou R7 = 0AH, en base 16.

4.4 Les registres R8 à R10

L'amplitude du signal sonore émis par les trois convertisseurs D/A (un pour chacun des canaux A, B et C) est déterminé par le contenu du registre R8 pour le canal A, du registre R9 pour le canal B et du registre R10 pour le canal C.

Seuls les 4 bits les moins significatifs (B3-B0) de chacun des registres sont utilisés. Ils permettent donc des valeurs comprises entre 0 et 15. La valeur 0 chargée dans l'un des registres correspond à un volume (nul). L'amplitude maximale est obtenue en chargeant le registre avec la valeur 15. Le cinquième bit (B4) est utilisé pour sélectionner le mode de fonctionnement du contrôle d'amplitude.

Si le bit B4 est égal à 0, l'amplitude ne varie pas.

Si le bit B4 est égal à 1, la variation d'amplitude est confiée au générateur d'enveloppe.

Registre de contrôle d'amplitude :

B7 B6 B5	B4	B3 B2 B1 B0
non utilisé	Mode	Niveau d'amplitude (4 bits)

4.5 Les registres R11 et R12

Les registres R11 et R12 permettent au PSG de contrôler la période de l'enveloppe. Un calcul similaire à celui des registres R0 à R5 fournit les valeurs à charger dans les registres R11 et R12.

Soit PE pour la période d'enveloppe et Fe pour la fréquence d'enveloppe :

PE = 3 579 545 / (256*Fe)

Les 8 bits des registres R11 et R12 sont utilisés. La valeur résultante est donc codée sur 16 bits et peut varier de 0 à 65535 (2^16 - 1).

Le registre R11 du PSG définit l'ajustement fin de la période d'enveloppe tandis que R12 permet un ajustement grossier.

Ajustement grossier R12	Ajustement fin R11
B7 B6 B5 B4 B3 B2 B1 B0	B7 B6 B5 B4 B3 B2 B1 B0
PE15 à PE8	PE7 à PE0

PE (16 bits pour le générateur d'enveloppe)

Par la formule, on peut calculer PE max et PE min des enveloppes possibles :

Fe = 3579545 / (256*PE)

Fe = 1/PE = 3579545 / (256*PE)

Fe = (256*PE) / 3579545

PE max = 65535 * 256 / 3579545 = 4,6868973 secondes

PE min = 1 * 256 / 3579545 = 0,0000715 seconde

4.6 Le registre R13

Le registre R13 contrôle les différentes formes de modulation utilisées par le PSG. Si le bit (B4) décrit dans les registres R8 à R10 est à 1, la modulation a lieu; sinon la programmation du registre 13 est ignorée. Seuls les 4 bits les moins significatifs sont utilisés. Ils déterminent chacun un paramètre de modulation.

Table des modulations :

BIT					Forme de l'enveloppe	Valeurs possibles
3	2	1	0			
0	0	X	X	A	Un seul cycle commence avec une amplitude maximum qui diminue pour devenir nulle	0, 1, 2, 3
0	1	X	X	B	Un seul cycle commence avec une amplitude nulle qui augmente pour atteindre sa valeur maximum, ensuite retombe brusquement à 0	4, 5, 6, 7
1	0	0	0	C	Comme A mais se répète sans cesse	8
1	0	1	0	D	Comme C, mais remonte de façon plus marquée vers le maximum (ATTACK)	10
1	0	1	1	E	Comme A mais revient ensuite au maximum et y reste	11
1	1	0	0	F	Comme B mais se répète sans cesse	12
1	1	0	1	G	Comme B mais reste au maximum	13
1	1	1	0	H	Comme F mais avec une attaque plus marquée	14

R13 : Contrôle de forme de modulation :

B7 B6 B5 B4 B3 B2 B1 B0 Fonction

\ / Hold Alternate Attack Continue

non utilisé

5 Utilisation des registres R0 à R13

La programmation peut se faire de deux façons différentes en Basic :

- soit en utilisant l'instruction SOUND
- soit en utilisant l'instruction OUT. Cette deuxième manière de procéder est également valable en assembleur.

La programmation d'un son au moyen de la commande SOUND est très aisée : il suffit d'écrire SOUND NR, Vl où NR est le numéro du registre (compris entre 0 et 13) et VL la valeur à écrire dans ce registre (comprise entre 0 et 255).

La programmation d'un son au moyen de l'instruction OUT est un peu plus compliquée : il faut donner le numéro du registre sur le port A0H (160) et ensuite écrire la valeur de VL sur le port A1H (161).

L'instruction SOUND NR, VL peut donc s'écrire OUT 160, NR : OUT 161, VL.

Programmes de démonstration :

- Génération d'une explosion (coup de feu)

```
10 SOUND 6, 15
20 SOUND 7, 7
30 SOUND 8, 16
40 SOUND 9, 16
50 SOUND 10, 16
60 SOUND 12, 16
70 SOUND 13,0
80 FOR I = 1 TO 500 : NEXT I
90 GOTO 70
```

Explication :

La ligne 10 détermine la fréquence du bruit (R6).

La ligne 20 valide la sortie du bruit sur les canaux A, B et C.

Les lignes 30 à 50 sélectionnent la modulation des 3 canaux au moyen du générateur d'enveloppe.

La ligne 60 détermine la période de l'enveloppe.

La ligne 70 détermine la forme de l'enveloppe (forme A) et produit le son. Chaque instruction SOUND 13, 0 reproduira le même son; c'est l'objet des lignes 80 (délai) et 90.

Au moyen de l'instruction OUT, le programme s'écrit :

```
10 LA = 160 : REM LA POUR LATCH ADRESS
15 WR = 161 : REM WR POUR WRITURE REGISTER
20 OUT LA, 6 : OUT WR, 15
30 OUT LA, 7 : OUT WR, 7
40 OUT LA,8 : OUT WR, 16
50 OUT LA,9 : OUT WR, 16
60 OUT LA,10 : OUT WR, 16
70 OUT LA,12 : OUT WR, 16
80 OUT LA,13 : OUT WR, 0
90 FOR I = 1 TO 500 : NEXT I
95 GOTO 80
```

- Génération d'un bruit de sirène :

```
10 SOUND 7,62 : REM SON CANAL A ON
20 SOUND 8,15 : REM VOLUME MAX SUR CANAL A
30 FOR J = 1 TO 3
40 FOR I = 100 TO 200 : REM VALEUR DE LA FRÉQUENCE
50 SOUND 0, I
60 NEXT I
70 FOR I = 200 TO 100 STEP -1 : REM ON REDESCEND
80 SOUND 0, I
90 NEXT I
95 NEXT J : REM ON FAIT 3 FOIS PIN-PON
99 OUT 8, 0 : REM AMPLITUDE À 0 ARRÊT DU SON
```

6 Utilisation des ports d'entrée/sortie

Nous avons vu, lors de l'étude du registre R7 que les bits 7 et 6 règlent le sens de la transmission des données (0 = entrée, 1 = sortie). Les registres R14 et R15 sont utilisés pour l'écriture et la lecture de ces deux ports.

6.1 Ecriture et lecture des ports

L'écriture dans les ports d'entrée/sortie se fait exactement comme l'écriture dans un registre du générateur sonore, mais l'instruction Basic SOUND n'est plus utilisable. Il faut donc utiliser l'instruction OUT.

Les deux ports sont appelés ports A et B.

L'écriture dans le port A se fait en chargeant R14 avec la valeur à écrire par la suite d'instructions suivantes :

OUT 160, 14 : OUT 161, VL

L'écriture dans le port B se fait en chargeant R15 avec la valeur à écrire par la suite d'instructions suivantes :

OUT 160, 15 : OUT 161, VL

Ce qui a été dit au sujet de l'assembleur lors de l'étude de la programmation de R0 à R13 reste valable pour la programmation de R14 et R15.

la lecture du contenu d'un port (A ou B) se fait en chargeant le port 160 du microprocesseur avec 14 pour le port A ou 15 pour le port B puis en effectuant une lecture du port 162 du microprocesseur au moyen de la fonction Basic « INP ».

Exemple : lecture du contenu du port A :

```
OUT 160, 14 : X = INP(162)
```

A la suite de ces instructions, la variable X contient le contenu du port A.

6.2 Contenu des ports A et B du PSG

Le port A sert principalement à la lecture des joysticks ou manettes de jeu.

Les 4 bits de poids faible sont utilisés pour déterminer le sens de déplacement de la manette de jeu.

Le bit 0 passe à 0 si la manette est poussée vers le haut.

Le bit 1 " 0 " " " " " le bas.

Le bit 2 " 0 " " " " " la gauche.

Le bit 3 " 0 " " " " " la droite.

Le bit 4 détecte la pression sur le bouton de tir.

Le bit 5 est utilisé pour armer le TRIGGER pour les manettes analogiques (paddles).

Le bit 6 n'est pas utilisé en version européenne.

Le bit 7 sert à la lecture des cassettes.

Le port B est principalement utilisé pour sélectionner la manette 1 ou la manette 2 ainsi que pour la gestion des manettes analogiques (bits 0 à 6). Le bit 7 n'est pas utilisé en version européenne.

6.3 Routines en assembleur

Il existe des routines en assembleur qui permettent la lecture des manettes logiques, des boutons de tir et des manettes analogiques.

Lecture des manettes logiques :

Il suffit de mettre dans A le nombre (1 ou 2) correspondant au numéro de la manette ou 0 pour le clavier et de faire un CALL à l'adresse 00D5H. La valeur du joystick est rendue dans A.

Lecture du bouton de tir :

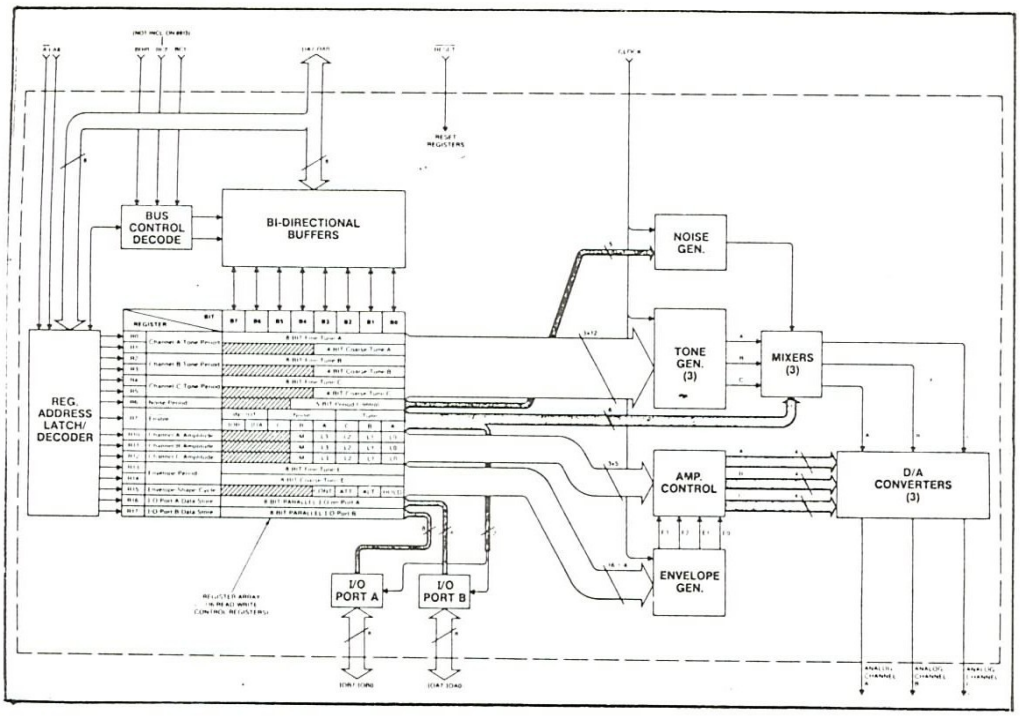
Il suffit de mettre dans A le numéro de manette et de faire un CALL à l'adresse 00D8H.

Lecture des manettes analogiques :

Deux routines existent :

L'une en 00DBH qui réalise l'équivalent de la fonction PAD.

L'autre en 00DEH qui réalise l'équivalent de la fonction PDL.



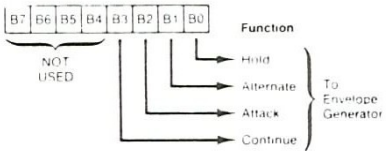


Table des modulations.

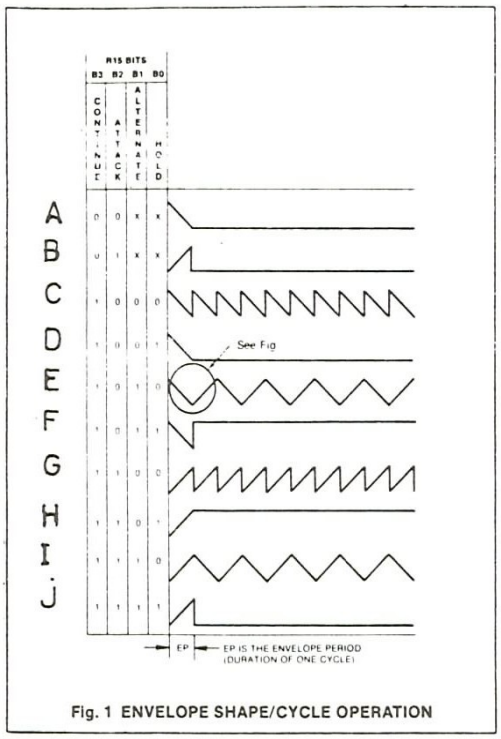


Fig. 1 ENVELOPE SHAPE/CYCLE OPERATION

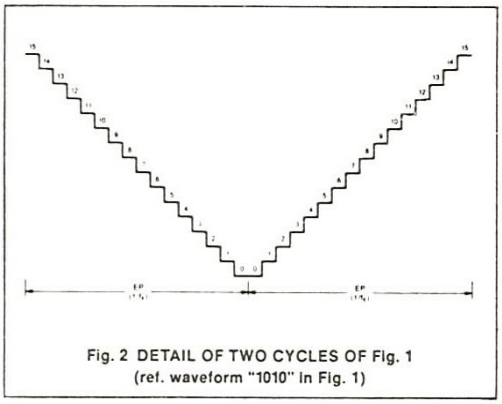


Fig. 2 DETAIL OF TWO CYCLES OF Fig. 1
(ref. waveform "1010" in Fig. 1)

CHAPITRE 3

LE PROCESSEUR DE GESTION D'ÉCRAN VDP 9938

1 Généralités

Le processeur de gestion d'écran est un circuit spécialement développée pour le MSX2. Ce circuit porte le nom de VDP 9938 (Video Display Processor). Il contient plus de circuits élémentaires que le processeur central Z80 et permet d'effectuer des fonctions d'une grande complexité en un temps minimum.

Le VDP 9938 est compatible avec son ancêtre le 9918 qui équipait les MSX1. Il autorise cependant un grand nombre d'améliorations :

- affichage de texte en 80 colonnes
- 256 couleurs
- 8 sprites par ligne horizontale
- 512 x 424 points possibles (non utilisé en MSX2)
- Accès direct en utilisant des coordonnées X et Y
- Fonctions graphiques élémentaires intégrées
- Gestion intégrée de la souris et du crayon optique
- Possibilité de digitalisation et d'incrustation d'images
- Synchronisation externe possible
- Sortie RGB et composite directe sur le circuit
- Gestion de 128 Ko de mémoire vive

1.1 Structure interne et tables

Ce circuit gère tous les signaux de contrôle nécessaires à la génération d'une image vidéo. Il s'occupe aussi du rafraîchissement de sa propre mémoire et du rappel de ces informations.

Il possède sa propre mémoire : elle n'occupe aucun espace sur le bus du processeur principal et par conséquent, la place réservée aux programmes est plus étendue.

Ceci représente un avantage par rapport aux machines dont la mémoire d'écran est comprise dans la mémoire centrle (ATARI, COMMODORE, TRS 80 mod I et II...).

Toutes les roses possédant des épines, voyons l'inconvénient. Si la mémoire n'est pas sur le bus, elle ne peut être atteinte directement et le jeu d'instructions se réduit de façon notable.

En effet, pour écrire dans la mémoire du VDP encore appelée vidéoram, on ne peut se servir que des instructions IN(P) et OUT.

Le Basic contient deux instructions qui permettent d'écrire et de lire dans la vidéoram comme POKE et PEEK permettent de lire et d'écrire dans la mémoire centrale. Ces deux instructions sont VPOKE et VPEEK : l'interpréteur Basic les transforme en IN et OUT.

Le VDP affiche une image sur l'écran qui peut être considérée comme un ensemble de plans semi-transparents disposés les uns derrière les autres.

Les objets qui se trouvent sur les plans les plus proches de la personne qui regarde l'écran ont la

plus grande priorité. Dans le cas où deux ou plusieurs objets occupent le même emplacement sur l'écran mais pas sur le même plan, l'objet ayant la plus haute priorité est visualisé.

Les 32 premiers plans peuvent contenir un et un seul sprite ou lutin (un sprite est un objet matricé sur 8x8, 16x16 ou 32x32 points qui est défini par ses coordonnées verticales et horizontales dans la vidéoram).

Les points du plan non occupés par le sprite sont transparents. Le sprite étant défini simplement par ses coordonnées, le déplacer dans l'écran ne présente aucune difficulté.

Derrière les 32 plans réservés aux sprites on trouve le plan réservé aux textes et aux graphiques normaux. Ensuite, on trouve le plan de couleur de fond : ce plan est plus large que les autres car il forme un bord autour des autres plans.

Les 32 plans réservés aux sprites sont disponibles en mode graphique ou multicolore. Ils ne sont pas utilisables en mode texte (mode 0) où ils sont automatiquement transparents.

Chacun des sprites peut recouvrir une zone de 8x8, 16x16 ou 32x32 points sur le plan. Chaque partie d'un plan non occupée par un sprite est transparente. Chaque point du sprite peut être transparent ou non. Le sprite 0 est le sprite le plus prioritaire. Quand un point est transparent, la couleur du point suivant est visualisée et s'il n'est pas transparent, les couleurs des plans suivants sont remplacés par la couleur du point en question.

La vidéoram peut être décomposée en une série de tables dont la taille et l'adresse varient en fonction du mode d'affichage.

La table des noms de patrons (TNP)

En mode texte, c'est un bloc de 960 mémoires contigües qui représente les 960 positions dans l'écran (40x24), chaque mémoire contient le numéro de caractère à afficher à cette position dans l'écran.

La première mémoire contient la valeur du caractère à afficher en haut et à gauche de l'écran, la 40ème contient la valeur du caractère à afficher en haut à droite de l'écran et bien sûr, la dernière (960) contient la valeur du caractère à afficher en bas et à droite de l'écran.

La table génératrice des patrons (TGP)

En mode texte, cette table contient le dessin des 256 caractères affichables. Elle est chargée en standard avec les caractères alphanumériques et semi-graphiques que vous connaissez. Ces caractères peuvent être modifiés par un programme. Chaque caractère est défini par une matrice de 6x8 points programmés sur 8 octets.

La matrice constituée de 6 points en horizontal sur 8 points en vertical est structurée comme suit : chaque ligne horizontale est programmée sur un octet dont seuls les 6 bits les plus significatifs sont utilisés : le bit le plus significatif définit le point le plus à gauche de l'horizontale.

La TGP permet de définir 256 caractères codés sur 8 octets. En mode texte, la taille de cette table est de 2048 octets.

En mode multicolore, la TGP contient l'information couleur pour chaque patron.

Dans les modes graphiques, la TGP définit l'état de chaque point.

La table des couleurs (TC)

Elle est utilisée dans les modes graphiques.

En mode graphique 1, la TC définit la couleur de chaque groupe de 8 patrons. Un octet est réservé par groupe : 32 octets sont donc nécessaires.

Chaque octet est divisé en 2x4 bits. Les 4 bits les plus significatifs (les plus à gauche) définissent la couleur des bits à 1 dans la TGP. Les 4 bits les moins significatifs (les plus à droite) définissent la couleur des bits à 0 dans la TGP. 4 bits permettant le codage de 16 couleurs, le compte y est.

Voici l'équivalence entre les bits et la couleur :

0000	(00)	- TRANSPARENT	1000	(08)	- ROUGE MOYEN
0001	(01)	- NOIR	1001	(09)	- ROUGE CLAIR
0010	(02)	- VERT MOYEN	1010	(10)	- JAUNE FONCE
0011	(03)	- VERT CLAIR	1011	(11)	- JAUNE CLAIR
0100	(04)	- BLEU FONCE	1100	(12)	- VERT FONCE
0101	(05)	- BLEU CLAIR	1101	(13)	- MAGENTA
0110	(06)	- ROUGE FONCE	1110	(14)	- GRIS
0111	(07)	- CYAN	1111	(15)	- BLANC

En mode graphique 2, la TC occupe 6144 octets. Elle permet alors de définir 2 couleurs pour chaque octet de la TGP, c'est-à-dire 2 couleurs pour chaque série de 8 points horizontaux.

La table génératrice des sprites (TGS)

Cette table contient le dessin du sprite. Le sprite est défini sur 8x8 ou 16x16 bits, cette valeur étant définie dans le registre 1.

Les bits à 1 sont affichés dans la couleur du sprite.

Les bits à 0 sont toujours transparents.

La table des attributs des sprites (TAS)

Cette table contient 4 valeurs pour chaque sprite. La première valeur est la position verticale du sprite sur un octet; la seconde valeur est la position horizontale du sprite sur un octet; la troisième valeur est un pointeur vers la TGS qui définit le dessin du sprite; la quatrième valeur définit enfin la couleur du sprite.

Comme vous disposez de 32 sprites composés de 4 octets chacun, 128 octets suffisent pour définir la TAS.

Des informations supplémentaires sur la TGS et la TAS vous seront fournies lors de l'étude des sprites.

2 Mécanisme d'adressage

2.1 Généralités

Le VDP est programmé par l'intermédiaire de registres. Ces registres seront décrits en détail dans la section suivante. Le VDP est aussi en contact avec une mémoire qui lui est exclusivement réservée appelée vidéoram ou VRAM.

Cette section a pour but de décrire les différentes techniques utilisées pour accéder aux registres ou à la mémoire du VDP.

L'interfaçage du VDP avec le monde extérieur est réalisé par l'intermédiaire de 2 adresses qui permettent de définir 4 ports d'entrée/sortie.

Ces 4 ports sont numérotés de 0 à 3 (nous les appellerons PORT VDP) et sont associés à des adresses de PORT du Z80 (nous les appellerons PORT MSX). Les adresses de ces ports MSSSX

peuvent varier. En principe, elles sont situées de 98H à 9BH. De toutes façons, l'adresse du premier de ces ports (98H en principe) est située aux octets 6 et 7 de la ROM Bios. Une lecture de la valeur de ces octets vous assurera une compatibilité parfaite.

2.2 Accès direct aux registres

Pour accéder directement un registre, il suffit de suivre la séquence suivante :

- écrire la donnée à mettre dans le registre sur le port 1 du VDP (interfacé par le port 99H du MSX)
- écrire le numéro de registre augmenté de 128 (80H ou bit 7 à 1) sur le même port.

Exemple : En Basic l'écriture de la valeur 64 dans le registre 1 se fait par l'instruction suivante :

OUT &H99, 64 : OUT &H99, 129 : REM 128 + 1

2.3 Accès indirect aux registres

Dans l'appel indirect, le numéro de registre à accéder est contenu dans un autre registre (R17).

Le registre R17 (qui contient le registre à accéder) doit être positionné par le mode direct.

Il suffit alors d'envoyer la (ou les) donnée(s) sur le port 3 du VDP qui est associé au port 9BH du MSX.

Le bit 7 de R17 (à 1) permet une auto-incrémentation du numéro de registre à chaque écriture. Cette méthode est utile pour accéder des registres consécutifs, notamment lors de l'utilisation des registres de commande (R32 à R46).

Exemple : Si l'on veut positionner 12 dans R36; 14 dans R37, 18 dans R38 et 22 dans R39, il suffit de mettre 164 (32 + 128, b7 à 1) dans R17 en mode direct, et d'envoyer successivement 12, 14, 18 et 22 dans le port 3 du VDP (port 9BH du MSX).

2.4 Accès aux registres de palette

Pour envoyer des données dans les registres de palette, vous devez mettre le numéro du registre de palette dans le registre R16 (au moyen de l'adressage direct) et envoyer 2 octets de données (dans le bon ordre) dans le port 2 du VDP (le port 2 du VDP est associé au port 9AH du MSX).

Les deux octets ont le format suivant :

octet 1 : 0 R2 R1 R0 0 B2 B1 B0 octet 2 : 0 0 0 0 0 V2 V1 V0

V0 à V2, B0 à B2 et R0 à R2 correspondent respectivement à la valeur de la composante verte, bleue et rouge de la couleur (8 possibilités par composante).

Exemple en assembleur : positionnement du registre de palette 12 dans la couleur blanche

```
LD    A, 0CH      ; registre palette 12
OUT   (99H), A    ; écriture port 99
LD    A, 90H      ; registre 16 (10H) + 128
OUT   (99H), A    ; écriture directe
LD    A, 77H      ; composantes bleues et rouges valeur max (7)
OUT   (9AH), A    ; premier octet envoyé
LD    A, 07H      ; composante verte valeur max (7)
OUT   (9AH), A    ; deuxième octet envoyé
```

2.5 Accès aux registres d'état

Pour accéder en lecture aux registres d'état (S0 à S9), il suffit de positionner le numéro du registre d'état à lire dans le registre R15 (en mode direct) et de lire le contenu du port 1 du VDP (le port 1 du VDP est associé au port 99H du MSX).

2.6 Accès à la vidéoram

Le VDP peut être connecté à une vidéoram (VRAM) de 128 Ko et être en relation avec une partie de la mémoire d'extension de 64 Ko.

L'accès à la mémoire se fait de la façon suivante :

- positionnement de R45 bit 6 pour déterminer le type d'accès (0 = VRAM, 1 = RAM EXTENSION)
- positionner l'adresse haute à accéder (A14 à A16) dans R14 (bits 0 à 2)
- positionner l'adresse basse (A0 à A7) à accéder en accédant le port 1 du VDP (por t 99H du MSX)
- positionner l'adresse moyenne (A8 à A13, bit B0 à B5) ainsi que le mode choisi (écriture ou lecture) en accédant le port 1 du VDP (port 99H du MSX). Le mode écriture ou lecture est déterminé par le bit 6 (0 = lecture, 1 = écriture). Le bit 7 est à 0.
- Lire la donnée sur le port 0 du VDP (port 98H du MSX)

Remarques :

- Comme le compteur d'adresses est automatiquement incrémenté après chaque accès, vous pouvez accéder à un grand nombre de données successivement en accédant le port 0 (port 98H du MSX) sans reprogrammer l'adresse.
- Respectez impérativement l'ordre des opérations !

2.7 Accès en Basic

Le Basic possède une série d'instructions permettant l'accès direct aux registres et à la vidéoram.

Les instructions VPEEK et VPOKE permettent d'accéder directement à la vidéoram en lecture (VPEEK) et en écriture (VPOKE).

L'instruction VDP permet de lire et de modifier le contenu d'un registre. Les registres de contrôle étant des registres à écriture seule, comment le Basic peut-il en déterminer le contenu ? EN vérité il ne le peut pas et les programmeurs de la ROM Basic ont utilisé une astuce : lors de chaque mise à jour interne (ou par l'intermédiaire de l'instruction VDP) d'un registre, la valeur écrite dans ce registre est copiée dans une partie de la mémoire de travail du Basic. C'est cette valeur qui est relue lors de la lecture de la valeur d'un registre. Evidemment, si la mise à jour se fait par un programme assembleur extérieur ou par une série d'instructions OUT, la valeur lue devient fausse et ne correspond plus à rien. Ces valeurs sont sauvées en mémoire aux adresses F3DF à F3E6 (R0 à R7) et FFE7 à FFF6 (R8 à R24).

Remarque : Différentes routines en assembleur pour manipuler les registres du VDP existent dans la ROM Bios, elles seront décrites dans le chapitre réservé au logiciel interne.

3 Les registres du VDP

Les registres qui équipent le VDP 9938 sont des registres 8 bits, ils sont de deux types :

- les registres de contrôle qui sont au nombre de 39 (0 à 23 et 32 à 46) et qui sont des registres à écriture seule. Nous les appellerons R0 à R23 et R32 à R46.
- Les registres d'état au nombre de 10 (0 à 9) et qui sont des registres à lecture seule. Nous les appellerons S0 à S9 (Status Register).

3.1 Les registres de contrôle

Ils peuvent être divisés en 6 groupes :

- les registre de mode (R0, R1, R8 et R9)
- Les registres d'adresse de base des tables (R2, R3, R4, R5, R6, R10 et R11)
- Les registres de couleur (R7, R12, R13, R20, R21 et R22)
- Les registres d'affichage (R18, R19 et R23)
- Les registres d'accès (R14, R15, R16 et R17)
- Les registres de commande (R32 à R46)

Le registre R0 : registre de mode 0

Bit	7	6	5	4	3	2	1	0
R0	0	DG	IE2	IE1	M5	M4	M3	0

- M3, M4 et M5 servent à modifier le mode d'affichage (avec M1 et M2 de R1).

M5	M4	M3	M2	M1	Mode
0	0	0	0	0	Graphique 1
0	0	1	0	0	Graphique 2
0	1	0	0	0	Graphique 3
0	1	1	0	0	Graphique 4
1	0	0	0	0	Graphique 5
1	0	1	0	0	Graphique 6
1	1	1	0	0	Graphique 7
0	0	0	0	1	TEXT1 (40)
0	1	0	0	1	TEXT2 (80)
0	0	0	1	0	Multicolore

- IE1 à 1 autorise les interruptions en provenance du balayage horizontal (elles permettent de réaliser des mises à jour de l'écran pendant le temps de retour du spot de balayage).
- IE2 à 1 autorise les interruptions en provenance du crayon optique.
- DG positionne le bus couleur en entrée et permet d'entrer des données dans la vidéoram.

Le registre R1 : registre de mode 1

Bit	7	6	5	4	3	2	1	0
R1	0	BL	IE0	M1	M2	0	SI	MAG

- MAG à 1 indique que la taille des sprites doit être multipliée par 2.
- SI à 1 indique que la taille des sprites est de 16x16 points. A 0, la taille des sprites est limitée à 8x8 points.
- M1 et M2 servent à définir le mode écran (avec M3, M4 et M5, voir R0).
- IE0 à 1 autorise les interruptions en provenance du balayage horizontal.
- BL à 1 autorise l'affichage, à 0 il n'y a pas d'affichage.

Le registre R2 : registre d'adresse de base le la TNP

Bit	7	6	5	4	3	2	1	0
R2	0	A16	A15	A14	A13	A12	A11	A10

A10 à A16 correspondent aux adresses eA10 à A16 de la vidéoram. LA TNP peut donc être installée n'importe où dans les 128 Ko de vidéoram par pas de 1 Ko.

Le registre R3 : registre d'adresse de base de la TC (partie basse)

Bit	7	6	5	4	3	2	1	0
R3	A13	A12	A11	A10	A9	A8	A7	A6

(voir R10)

Le registre R4 : registre d'adresse de base de la TGP

Bit	7	6	5	4	3	2	1	0
R4	0	0	A16	A15	A14	A13	A12	A11

Le registre R4 permet de disposer la TGP (Table Génératrice des Patrons) n'importe où dans les 128

Ko de vidéoram par pas de 2 Ko.

Le registre R5 : registre d'adresse de base de la TAS (partie basse)

Bit	7	6	5	4	3	2	1	0
R5	A14	A13	A12	A11	A10	A9	A8	A7

Utilisé conjointement avec R11, le registre R5 permet de disposer la TAS (Table d'Attributs des Sprites) n'importe où dans les 128 Ko de la vidéoram par pas de 128 octets.

Le registre R6 : registre d'adresse de base de la TGS

Bit	7	6	5	4	3	2	1	0
R6	0	0	A16	A15	A14	A13	A12	A11

Le registre R6 permet de disposer la TGS (Table Génératrice des Sprites) n'importe où dans les 128 Ko de vidéoram par pas de 2 Ko.

Le registre R7 : registre de couleur TEXTE et FOND

Bit	7	6	5	4	3	2	1	0
R7	TC3	TC2	TC1	TC0	BD3	BD2	BD1	BD0

- TC0 à TC3 sélectionnent la couleur du texte en mode TEXT1 et en TEXT2 (16 possibilités).
- BD0 à BD3 sélectionnent la couleur du fond dans tous les modes.

Le registre R8 : registre de mode 2

Bit	7	6	5	4	3	2	1	0
R8	MS	LP	TP	CB	VR	0	SPD	BW

- BW à 1 sélectionne le mode noir et blanc (32 niveaux de gris), à 0 sélectionne le mode couleur.
- SPD à 1 empêche l'affichage des sprites, à 0 il l'autorise.
- VR sélectionne le type de vidéoram (16 Ko ou 64 Ko). Vous ne devez à aucun prix modifier ce bit.
- CB à 1 sélectionne le bus couleur en entrée, à 0 sélectionne le bus couleur en sortie.
- TP à 1 positionne la couleur de code 0 comme couleur de la palette.
- LP à 1 autorise le crayon optique, 0 à il l'inhibe.
- MS à 1 positionne le bus en mode entrée et autorise la souris, à 0 positionne le bus en mode

sortie et inhibe la souris.

Le registre R9 : registre de mode 3

Bit	7	6	5	4	3	2	1	0
R9	LN	0	S1	S0	IL	E0	*NT	DC

- DC à 1 positionne DLCK en mode entrée, à 0 positionne DLCK en mode sortie.
- *NT à 1 sélectionne le mode PAL (313 lignes), à 0 sélectionne le mode NTSC (262 lignes). Ne modifiez jamais ce bit.
- E0 à 1 affiche 2 écrans graphiques interchangeables, à 0 affiche deux fois le même écran graphique.
- IL à 1 sélectionne le mode entrelacé, à 0 sélectionne le mode non entrelacé. Ne modifiez jamais ce bit.
- S0 et S1 sélectionnent le mode de synchronisation (voir paragraphe 7.4)
- LN à 1 sélectionne 212 lignes horizontales, à 0 sélectionne 192 lignes horizontales.

Le registre R10 : registre d'adresse de base de la TC (partie haute)

Bit	7	6	5	4	3	2	1	0
R10	0	0	0	0	0	A16	A15	A14

Utilisé conjointement avec R3, R10 permet de disposer la TC (Table des Couleurs) n'importe où dans les 128 Ko de vidéoram par pas de 64 octets.

Le registre R11 : registre d'adresse de base de la TAS (partie haute)

Bit	7	6	5	4	3	2	1	0
R11	0	0	0	0	0	0	A16	A15

Utilisé conjointement avec R5, le registre R11 permet de disposer la TAS (Table d'Attributs des Sprites) n'importe où dans les 128 Ko de la vidéoram par pas de 128 octets.

Le registre R12 : registre de couleur TEXTE et FOND

Bit	7	6	5	4	3	2	1	0
R12	T23	T22	T21	T20	BC3	BC2	BC1	BC0

- T20 à T23 sélectionnent la couleur du texte en mode TEXT2 lorsqu'il y a utilisation du clignotement.
- BC0 à BC3 sélectionnent la couleur du fond.

La couleur choisie par ce registre est affichée alternativement avec celle choisie par le registre R7 lors du clignotement.

Le registre R13 : registre du temps de clignotement

Bit	7	6	5	4	3	2	1	0
R13	ON3	ON2	ON1	ON0	OF3	OF2	OF1	OF0

Dans les modes graphiques 4 à 7, deux pages peuvent être affichées alternativement. Le registre R13 permet de déterminer le temps d'apparition de chacune des images.

- ON0 à ON3 sélectionnent la durée d'apparition de l'image 1.
- OF0 à OF3 sélectionnent la durée d'apparition de l'image 2.

Table des durées :

B3	B2	B1	B0	Durée (ms)
0	0	0	0	0
0	0	0	1	166,9
0	0	1	0	333,8
0	0	1	1	500,6
0	1	0	0	667,5
0	1	0	1	834,4
0	1	1	0	1001,3
0	1	1	1	1168,2
1	0	0	0	1335,1
1	0	0	1	1501,9
1	0	1	0	1668,8
1	0	1	1	1835,7
1	1	0	0	2002,6
1	1	0	1	2169,5
1	1	1	0	2336,3
1	1	1	1	2503,2

Le registre R14 : registre d'accès à la VRAM

Bit	7	6	5	4	3	2	1	0
R14	0	0	0	0	0	A16	A15	A14

Lors de l'accès aux registres de la vidéoram (VRAM), le registre R14 contient les 3 bits de poids fort de l'adresse à accéder dans la VRAM.

Si une opération sur la VRAM génère une adresse supérieure, le report est automatiquement fait dans R14 sauf en mode graphique 1 et 2, TEXT1 et Multicolore (les modes du MSX1).

Le registre R15 : registre d'accès aux registres d'état

Bit	7	6	5	4	3	2	1	0
R15	0	0	0	0	S3	S2	S1	S0

Lors de la lecture d'un registre d'état, ce registre doit contenir le numéro du registre à lire (S0 à S3 pour 0 à 9).

Le registre R16 : registre d'accès à la palette couleur

Bit	7	6	5	4	3	2	1	0
R16	0	0	0	0	C3	C2	C1	C0

Ce registre doit contenir le numéro de la palette à sélectionner.

Le registre R17 : registre d'accès aux registres de contrôle

Bit	7	6	5	4	3	2	1	0
R17	AII	0	RS5	RS4	RS3	RS2	RS1	RS0

Ce registre permet d'accéder un autre registre de façon indirecte.

- RS0 à RS5 contiennent le numéro de registre à accéder.
- AII à 1 autorise l'auto-incrémentation des données, à 0 inhibe l'auto-incrémentation des données.

Le registre R18 : registre d'affichage et d'ajustement

Bit	7	6	5	4	3	2	1	0
R18	V3	V2	V1	V0	H3	H2	H1	H0

- V0 à V3 permettent l'ajustement vertical de l'écran.
- H0 à H3 permettent l'ajustement horizontal de l'écran.

Les valeurs 1 à 7 correspondent au déplacement vers la gauche ou vers le bas.
Les valeurs 8 à 15 correspondent au déplacement vers la droite ou vers le haut.
La valeur 0 correspond au centrage de l'image.

Le registre R19 : registre de sélection d'interruption

Bit	7	6	5	4	3	2	1	0
R19	IL7	IL6	IL5	IL4	IL3	IL2	IL1	IL0

IL 0 à IL7 permettent de déterminer le numéro de la ligne verticale qui produira une interruption.

Les registres R20 à R22 : registres de couleurs

Bit	7	6	5	4	3	2	1	0
R20	0	0	0	0	0	0	0	0
R21	0	0	1	1	1	0	1	1
R22	0	0	0	0	0	1	0	1

Ces trois registres sont chargés à l'initialisation du système et déterminent le signal couleur.

Le registre R23 : registre de décalage d'affichage

Bit	7	6	5	4	3	2	1	0
R23	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

DO0 à DO7 contiennent le numéro de la ligne où commence l'affichage.

Les registres R32 à R35 : registres sources X et Y

Bit	7	6	5	4	3	2	1	0

R32	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0
R33	0	0	0	0	0	0	0	SX8
R34	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0
R35	0	0	0	0	0	0	SY9	SY8

Ces registres contiennent la position X et la position Y de la source lors d'une commande interne. Reportez-vous à la section décrivant les commandes internes pour plus d'informations.

SX0 à SX8 peuvent prendre une valeur comprise entre 0 et 511.

SY0 à SY9 peuvent prendre une valeur comprise entre 0 et 1023.

Les registres R36 à R39 : registres destination X et Y

Bit	7	6	5	4	3	2	1	0
R36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0
R37	0	0	0	0	0	0	0	DX8
R38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0
R39	0	0	0	0	0	0	DY9	DY8

Ces registres contiennent la position X et la position Y de la destination lors d'une commande interne. Pour de plus amples informations, reportez-vous à la section dérivant les commandes internes.

DX0 à DX8 peuvent prendre une valeur comprise entre 0 et 511.

DY0 à DY9 peuvent prendre une valeur comprise entre 0 et 1023.

Les registres R40 à R43 : registres points X et Y

Bit	7	6	5	4	3	2	1	0
R40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0
R41	0	0	0	0	0	0	0	NX8
R42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0

R43	0	0	0	0	0	0	NY9	NY8

Ces registres contiennent le nombre de points en X et en Y sur lesquels portera la commande interne. Pour plus d'informations, reportez-vous à la section dérivant les commandes internes.

NX0 à NX8 peuvent prendre une valeur comprise entre 0 et 511.

NY0 à NY9 peuvent prendre une valeur comprise entre 0 et 1023.

Le registre R44 : registre de couleur

Bit	7	6	5	4	3	2	1	0
R44	CH3	CH2	CH1	CH0	CL3	CL2	CL1	CL0

Ce registre peut avoir plusieurs formats en fonction de la commande envoyée et du mode utilisé.

Le registre R45 : registre d'argument

Bit	7	6	5	4	3	2	1	0
R45	IL7	IL6	IL5	IL4	IL3	IL2	IL1	IL0

- MAJ : Dans la commande LINE (ligne), ce bit indique le sens du grand côté (0 = axe des X, 1 = axe des Y).
- EQ : Dans la commande SEARCH, ce bit indique le mode d'arrêt du SEARCH (0 = une couleur différente du bord, 1 = sur une couleur égale à celle du bord).
- DIX indique la direction en X : 0 = droite, 1 = gauche.
- DIY indique la direction en Y : 0 = bas, 1 = haut.
- MXS sélectionne la mémoire source : 0 = VRAM, 1 = RAM extension.
- MXD sélectionne la mémoire destination : 0 = VRAM, 1 = RAM extension.
- MXC n'est pas utilisé dans un MSX.

Le registre 46 : registre de commande

Bit	7	6	5	4	3	2	1	0
R46	CM3	CM2	CM1	CM0	LO3	LO2	LO1	LO0

- CM0 à CM3 indique le type de commande

CM3	CM2	CM1	CM0	Action
0	0	0	0	STOP
0	0	0	1	Non valide

0	0	1	0	Non valide
0	0	1	1	Non valide
0	1	0	0	POINT
0	1	0	1	PSET
0	1	1	0	SEARCH
0	1	1	1	LINE
1	0	0	0	Déplacement logique VDP → VRAM
1	0	0	1	Déplacement logique VRAM → VRAM
1	0	1	0	Déplacement logique VRAM → CPU
1	0	1	1	Déplacement logique CPU → VRAM
1	1	0	0	Déplacement rapide VDP → VRAM
1	1	0	1	Déplacement rapide VRAM → VRAM
1	1	1	0	Déplacement rapide VRAM → CPU
1	1	1	1	Déplacement rapide CPU → VRAM

– LO0 à LO3 indique l'opération logique à effectuer sur la couleur de destination.
DC = couleur de destination, DC/ = inverse logique de DC

SC = couleur de source, SC/ = inverse logique de SC

LO3	LO2	LO1	LO0	Action
0	0	0	0	IMP (implication) DC = SC
0	0	0	1	AND (et logique) DC = SC * DC
0	0	1	0	OR (ou logique) DC = SC + DC
0	0	1	1	EOR (ou exclusif) DC = SC/ * DC + SC * DC/
0	1	0	0	NOT (négation) DC = SC/
0	1	0	1	Non utilisé
0	1	1	0	Non utilisé
0	1	1	1	Non utilisé
1	0	0	0	TIMP Si SC = 0 DC = DC sinon DC = SC
1	0	0	1	TAND Si SC = 0 DC = DC sinon DC = SC * DC
1	0	1	0	TOR Si SC = 0 DC = DC sinon DC = SC + DC
1	0	1	1	TEOR Si SC = 0 DC = DC sinon DC = SC/ * DC + SC * DC/
1	1	0	0	TNOT Si SC = 0 DC = DC sinon DC = SC/
1	1	0	1	Non utilisé
1	1	1	0	Non utilisé
1	1	1	1	Non utilisé

3.2 Les registres d'état

Les dix registres d'état (S0 à S9) sont des registres utilisables en lecture uniquement. Ils permettent d'obtenir une série de renseignements sur l'état du VDP à un instant précis.

Le registre S0 : registre d'état 0

Bit	7	6	5	4	3	2	1	0
S0	F	5S	C	Numéro du 5ème sprite				

Les bits 0 à 4 indiquent le numéro du 9ème ou du 5ème sprite qui se trouve sur une même horizontale (les modes MSX1 acceptent 4 sprites, les modes MSX2 acceptent 8 sprites sur une même horizontale).

- C à 1 indique que 2 sprites sont entrés en collision.
- 5S à 1 indique que 5 sprites figurent sur la même horizontale.
- F sémaphore d'interruption du balayage vertical. Le bit est remis à 0 par une lecture du registre.

Le registre S1 : registre d'état 1

Bit	7	6	5	4	3	2	1	0
S1	FL	LPS	Numéro d'identification					FH

Les bits B1 à B5 fournissent le numéro d'identification du VDP.

- FH : sémaphore d'interruption de balayage vertical (voir R19). Si IE1 est mis à 1, l'interruption est autorisée. Lors de la lecture de S1, FH est remis à 0.
- LPS : sémaphore utilisé par le crayon optique et la souris. Si l'interrupteur du crayon ou le premier bouton de la souris est pressé, le sémaphore LPS passe à 1. Le bit n'est pas remis à 0 lors de la lecture du registre S1.
- FL : sémaphore utilisé par le crayon optique et la souris. Si le crayon détecte un point lumineux, le sémaphore FL passe à 1. Si le bit IE2 (R0 bit 5) est à 1, une interruption se produit. Lors de la lecture du registre S1, le bit est remis à 0. Si l'on appuie sur le deuxième bouton de la souris, le sémaphore FL passe à 1. Ce bit n'est pas remis à 0 par une lecture du registre S1.

Le registre S2 : registre d'état 2

Bit	7	6	5	4	3	2	1	0
S2	TR	VR	HR	BD	1	1	EO	CE

- CE : Sémaphore d'exécution. A l'état 1, il indique qu'une commande est en cours d'exécution.
- EO à 0 indique que le premier champ est affiché, à 1 indique que le second champ est affiché.
- BD sémaphore de détection de limite d'une couleur.
- HR sémaphore mis à 1 pendant le balayage horizontal.
- VR sémaphore mis à 1 lors du balayage vertical.
- TR sémaphore indiquant qu'un transfert entre le CPU et la VRAM peut être effectué.

Registres S3 à S6 : Registres X, Y

Bit	7	6	5	4	3	2	1	0
S3	X7	X6	X5	X4	X3	X2	X1	X0
S4	1	1	1	1	1	1	1	X8
S5	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
S6	1	1	1	1	1	1	1	Y8

Les registres S3 et S4 indiquent le numéro de colonne de la collision entre 2 sprites, de la localisation du crayon optique ou de la souris.

Les registres S5 et S6 indiquent le numéro de la ligne de la collision entre 2 sprites, de la localisation du crayon optique ou de la souris.

Registre S7 : registre d'indication de couleur

Bit	7	6	5	4	3	2	1	0
S7	C7	C6	C5	C4	C3	C2	C1	C0

Ce registre indique la valeur de la couleur rencontrée lors d'une commande POINT.

Registres S8 et S9 : registres de coordonnée de bord

Bit	7	6	5	4	3	2	1	0
S8	BX7	BX6	BX5	BX4	BX3	BX2	BX1	BX0
S9	1	1	1	1	1	1	1	BX8

Ces registres contiennent la coordonnée X de rencontre d'un bord d'une certaine couleur lors de la commande SEARCH.

4 Les registres et les tables dans les modes

4.1 Mode texte 1

Caractéristiques

Taille des caractères (patron) :	6x8 (en hauteur)
Nombre de caractères :	256
Taille écran :	40x24
Couleurs :	2 couleurs parmi 512
Taille mémoire d'un écran :	4 Ko

Contrôle

Table des caractères :	TGP dans la VRAM
Emplacement écran :	TNP dans la VRAM
Couleur caractère 1 :	4 bits haut de R7
Couleur caractère 2 :	4 bits bas de R7
Couleur de fond :	4 bits bas de R7

Sélection du mode (R0 & R1) :	M1	M2	M3	M4	M5
	1	0	0	0	0

Utilisation de la TGP : La TGP mémorise le patron de chaque caractère. Cette table est chargée de la ROM Bios vers la VRAM lors de l'initialisation du mode TEXT1.

La TGP contient 256 patrons numérotés de PN0 à PN255. Chaque patron occupe 8 octets. Dans chaque octet, les deux bits les moins significatifs ne sont pas affichés (6x8).

La TGP occupe donc 256 x 8 octets soit 2 Ko.

L'adresse de départ de la TGP est fixée par R4 par multiple de 2 Ko dans les 128 Ko de mémoire VRAM.

Exemple : patron de codage du caractère A dans la TGP :

Contenu	Octet
00100000	0
01010000	1
10001000	2
10001000	3
11111000	4
10001000	5
10001000	6
00000000	7

Utilisation de la TNP : la TNP est composée de 960 octets. Chaque octet représente une position de l'écran (24x40).

Chaque octet contient un numéro de patron de caractère qui est mis en correspondance avec la TGP.

L'adresse de départ de la TNP est disposée dans le registre R2 par multiple de 1 Ko dans les 128 Ko de mémoire VRAM.

Exemple : pour afficher un A (patron 65 de la TGP) sur la dixième colonne de la dernière ligne de

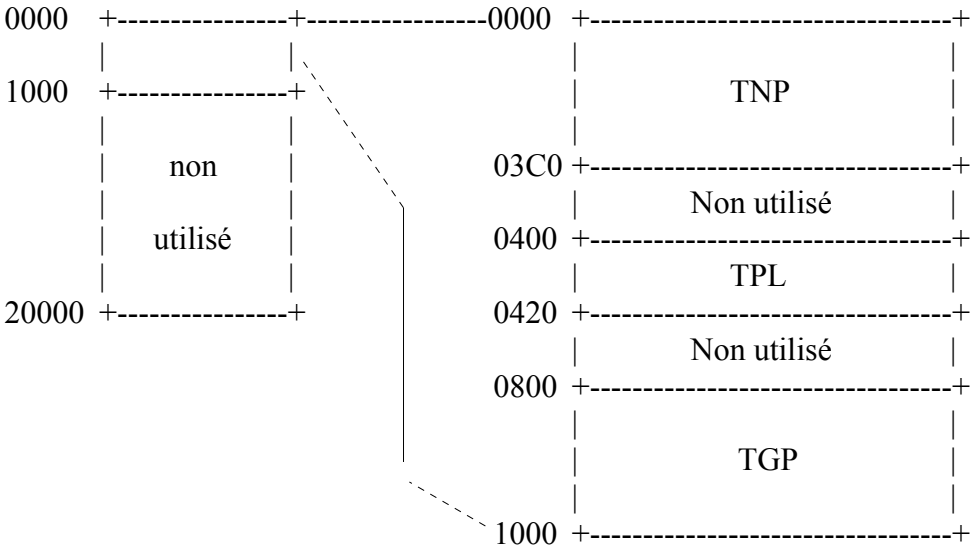
l'écran, il suffit d'écrire 65 dans l'octet relatif 929 de la TNP (40*23+9).

Remarques :

- L'ensemble TGP + TNP occupe 3 Ko et la TGP ne peut se disposer que par pas de 2 Ko dans la VRAM. Le mode TEXT1 nécessite donc 4 Ko (dont 2 Ko et 960 octets sont utilisés). Il est donc possible de définir 32 pages (128/4) en mode TEXT1.
- La couleur de fond est identique à la couleur 2 du caractère. Il ne peut donc y avoir que 2 couleurs en même temps sur l'écran.

STRUCTURE DES TABLES EN STANDARD

TNP	0000H - 03BFH	960 octets
TGP	0800H - 0FFFH	2048 octets
TPL	0400H - 041FH	32 octets



4.2 Mode texte 2

Caractéristiques

Taille des caractères (patron) :	6x8 (en hauteur)
Nombre de caractères :	256
Taille écran :	80x24 (192) ou 80x26,5 (212)
Couleurs :	2 couleurs parmi 512
	4 couleurs si clignotement
Taille mémoire d'un écran :	8 Ko

Contrôle

Table des caractères :	TGP dans la VRAM
Emplacement écran :	TNP dans la VRAM
Couleur caractère 1 :	4 bits haut de R7
Couleur caractère 2 :	4 bits bas de R7

Couleur de fond :	4 bits bas de R7
Clignotement :	TC dans la VRAM
Couleur caractère 1 :	4 bits hauts de R12 (en clignotement)
Couleur caractère 2 :	4 bits bas de R12 (en clignotement)

Sélection du mode (R0 & R1) :	M1	M2	M3	M4	M5
	1	0	0	1	0

LN (bit 7 de R9) permet de choisir entre 24 et 26,5 lignes.

Utilisation de la TGP : la TGP est exactement semblable à celle du mode TEXT1.

Utilisation de la TNP : La TNP est composée de 2160 octets maximum (en mode 26,5 lignes). Chaque octet représente une position de l'écran (27x80).

Chaque octet contient un numéro de patron de caractère qui est mis en correspondance avec la TGP.

L'adresse de départ de la TNP est disposée dans le registre R2 par multiple de 4 Ko dans les 128 Ko de mémoire VRAM. Les bits 0 et 1 de R2 doivent être à 1.

Utilisation de la TC : Chaque élément de la TNP est en relation avec 1 bit de la TC. Si ce bit est à 1, le caractère affiché à l'emplacement correspondant est affiché en clignotement.

Le bit le plus significatif de chaque octet de la TC correspond au caractère le plus à gauche de la TNP. La TC se compose donc de 270 caractères (2160/8). Le bit 7 du premier octet de la TC correspond au caractère en haut à gauche de l'écran.

La position de la TC est déterminée par le registre R3 (bits 3 à 7) et R10 (bits 0 à 2). Les bits 0 à 2 de R3 doivent être mis à 1, les bits 3 à 7 de R10 doivent être mis à 0.

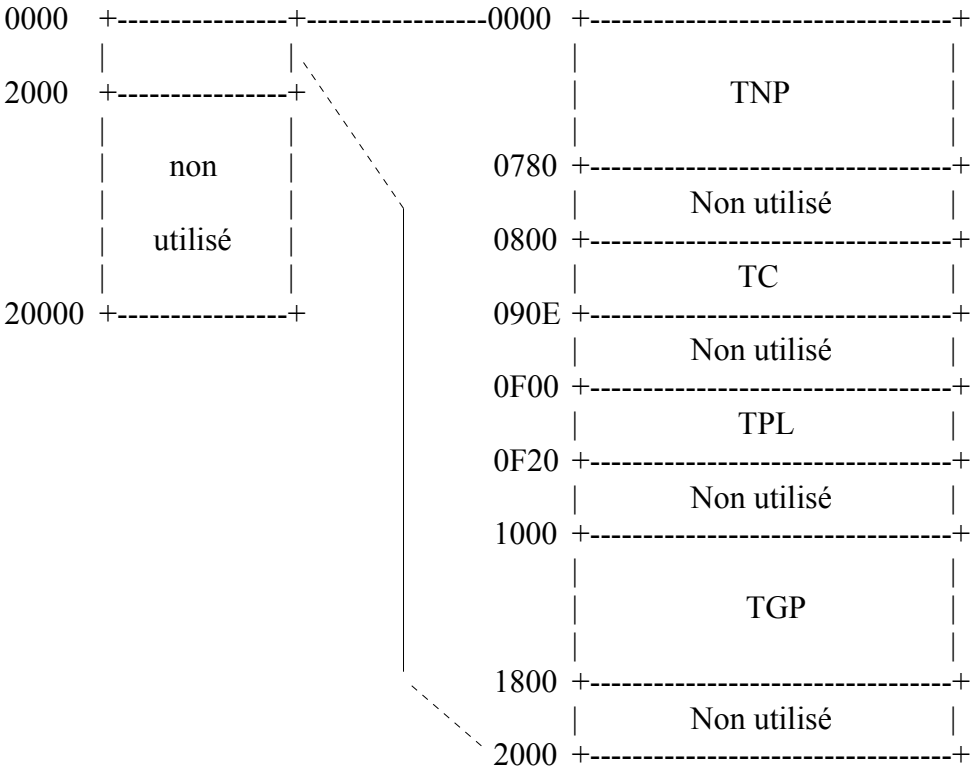
Les registres R7 et R12 permettent de définir la couleur du caractère en mode normal et en mode clignotement (voir description de ces registres).

Le registre R13 permet de déterminer la période de clignotement des caractères (voir description de ce registre).

Remarque : l'ensemble TGP + TNP + TC occupe 8 Ko. Il est donc possible de définir 16 pages (128/8) en mode TEXT2.

STRUCTURE DES TABLES EN STANDARD

TNP	0000H - 077FH	1920 octets
TGP	1000H - 17FFH	2048 octets
TC	0800H - 090DH	270 octets
TPL	0F00H - 0F1FH	32 octets



4.3 Mode multicolore

Caractéristiques

Taille écran :	64x48 blocs graphiques
Couleurs :	16 couleurs parmi 512
Mode sprite :	Mode 1
Taille mémoire d'un écran :	4 Ko

Contrôle

Table des couleurs blocs :	TGP dans la VRAM
Emplacement blocs :	TNP dans la VRAM
Couleur de fond :	4 bits bas de R7
Sprites :	TAS et TGS dans la VRAM

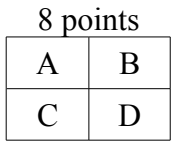
Sélection du mode (R0 & R1) :	M1	M2	M3	M4	M5
	0	1	0	0	0

Utilisation de la TGP : La TGP sert à mémoriser la couleur de chaque bloc. Chaque bloc a une taille de 4x4 points. Les blocs sont groupés par 4 (A, B, C et D) dans une matrice de 8x8 points. Il faut 4 bits pour définir une couleur parmi 16. Il faut donc 2 octets (16 bits) pour définir la couleur d'un groupe de 4 blocs.

Il y a 3072 blocs, il faut donc 1536 octets pour définir totalement la TGP. Cependant, la TGP est conçue de manière particulière. Elle est composée de 256 séries de groupes. Chaque série définit 4

groupes de 4 blocs. Le premier groupe est utilisé lorsque la coordonnée Y d'affichage du groupe est égale à 0, 4, 8, 12, 16 ou 20, le deuxième est utilisé lorsque la coordonnée d'affichage du groupe est égale à 1, 5, 9, 13,17 ou 21 et ainsi de suite. Les 4 groupes occupent donc 4x2 octets soit 8 octets. On retrouve ainsi une TGP de 2 Ko (256x8 octets).

R4 permet de positionner la TGP par pas de 2 Ko dans les 128 Ko de VRAM.



	B7 B6 B5 B4	B3 B2 B1 B0
octet 1	[couleur A]	[couleur B]
octet 2	[couleur C]	[couleur D]

Utilisation de la TNP : La TNP est composée d'un octet pour chaque groupe de 4 blocs. Il y a donc 768 octets dans la TNP. La valeur contenue dans un octet de la TNP est en relation avec une série de groupes de la TGP. C'est la position (Y) dans la TNP qui détermine celui des 4 groupes qui doit être utilisé.

L'adresse de départ de la TNP est disposée dans le registre R2 par multiple de 1 Ko dans les 128 Ko de mémoire VRAM.

Le registre R7 (partie basse) définit la couleur du fond.

R5 et R11 déterminent l'adresse de la TAS.

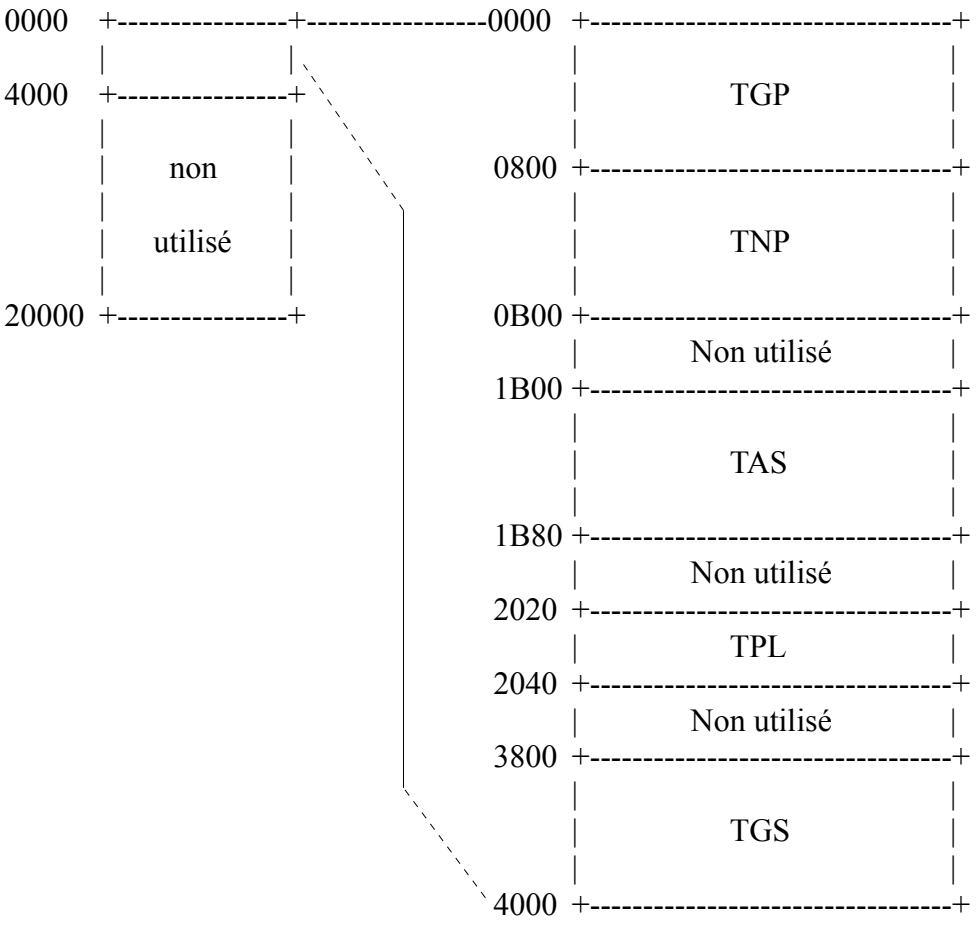
R6 détermine l'adresse de la TGS.

Reportez-vous à la section réservée aux sprites pour de plus amples informations.

Remarque : l'ensemble TGP + TNP occupe 4 Ko. Il est donc possible de définir 32 pages (128/4) en mode Multicolore.

STRUCTURE DES TABLES EN STANDARD

TGP	0000H - 07FFH	2048 octets
TNP	0800H - 0AFFH	768 octets
TAS	1B00H - 1B7FH	128 octets
TGS	3800H - 3FFFH	2048 octets
TPL	2020H - 203FH	32 octets



4.4 Mode graphique 1

Caractéristiques

Taille des matrices :	8x8
Nombre de matrices :	256
Taille écran :	32x24 matrices
Couleurs :	16 couleurs parmi 512
Mode sprite :	Mode 1
Taille mémoire d'un écran :	4 Ko

Contrôle

Table des matrices (dessin) :	TGP dans la VRAM
Emplacement matrices :	TNP dans la VRAM
Code de couleur matrice :	TC dans la VRAM
Couleur de fond :	4 bits bas de R7
Sprites :	TAS et TGS dans la VRAM

Sélection du mode (R0 & R1) :	M1	M2	M3	M4	M5
	0	0	0	0	0

Structure de la TGP : La TGP est constituée de 256 matrices de 8 octets. R4 détermine l'adresse de

départ de la TGP dans la VRAM par pas de 2 Ko.

Structure de la TNP : La TNP est composée de 768 octets (24x32). A chaque position de la TNP (qui correspond à une position sur l'écran) on peut faire correspondre une matrice de la TGP. Il suffit d'écrire dans l'octet correspondant dans la TNP le numéro de la matrice (0 à 255) dans la TGP. R2 détermine l'adresse de départ de la TNP dans la VRAM par pas de 1 Ko.

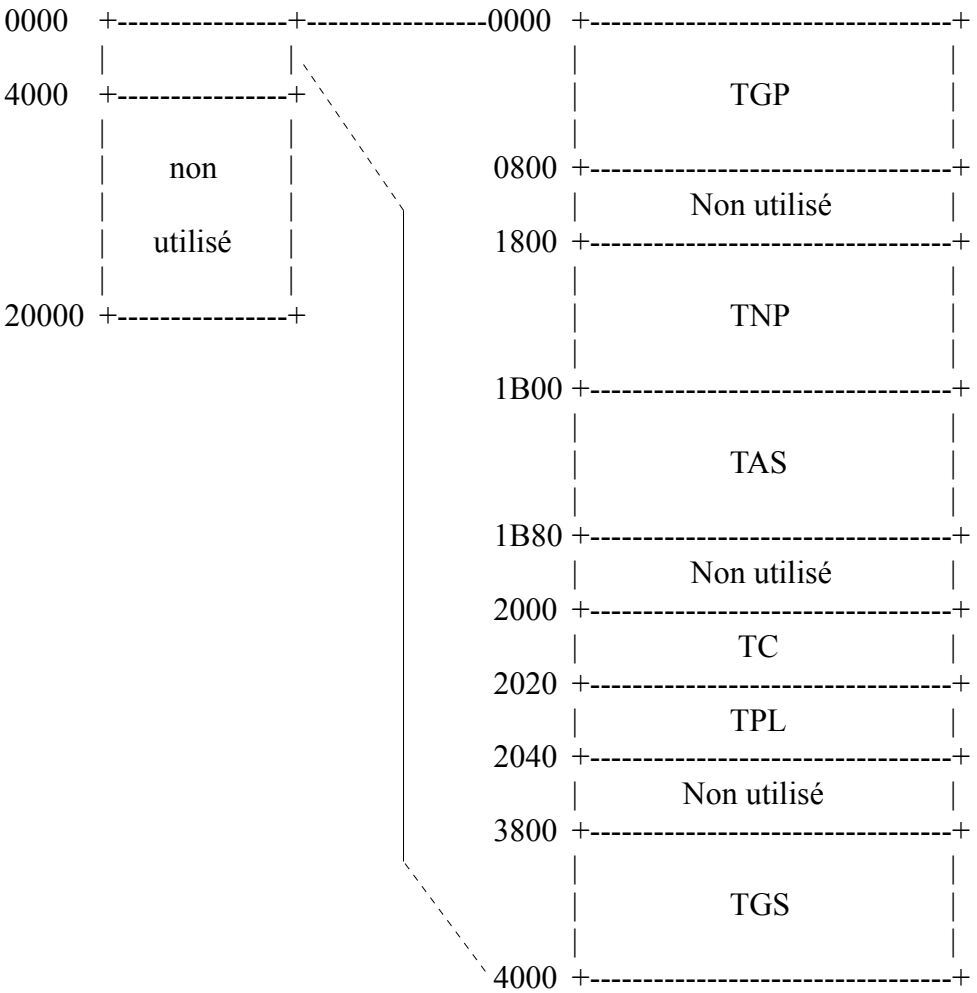
Structure de la TC : La TC se compose de 32 octets. Chaque octet détermine la couleur de 8 matrices consécutives dans la TGP. Ainsi, le premier octet de la TC correspond aux matrices 0 à 7 de la TGP. Le dernier octet de la TC correspond aux matrices 248 à 255 de la TGP. Chaque octet de la TC est divisé en 2 parties : les bits 0 à 3 déterminent la couleur des points à 0 dans la TGP, les bits 4 à 7 déterminent la couleur des points à 1 dans la TGP. R3 associé à R10 détermine l'adresse de départ de la TC dans la VRAM par pas de 32 octets.

La partie basse du registre R7 est utilisée pour définir la couleur du fond.

Les sprites fonctionnent en mode 1. R5 et R11 permettent de déterminer la position de la TAS. R6 permet de déterminer la position de la TGS.

STRUCTURE DES TABLES EN STANDARD

TGP	0000H - 07FFH	2048 octets
TNP	1800H - 1AFFH	768 octets
TC	2000H - 201FH	32 octets
TAS	1B00H - 1B7FH	128 octets
TGS	3800H - 3FFFH	2048 octets
TPL	2020H - 203FH	32 octets



4.5 Mode graphique 2

Caractéristiques

Taille des matrices :	8x8
Nombre de matrices :	768
Taille écran :	32x24 matrices
Couleurs :	16 couleurs parmi 512
Mode sprite :	Mode 1
Taille mémoire d'un écran :	16 Ko

Contrôle

Table des matrices (dessin) :	TGP dans la VRAM
Emplacement matrices :	TNP dans la VRAM
Code de couleur matrice :	TC dans la VRAM
Couleur de fond :	4 bits bas de R7
Sprites :	TAS et TGS dans la VRAM

Sélection du mode (R0 & R1) :	M1	M2	M3	M4	M5
	0	0	1	0	0

Structure de la TGP : La TGP est constituée de 768 matrices de 8 octets. Elle occupe donc 6 Ko. R4 détermine l'adresse de départ de la TGP dans la VRAM par pas de 8 Ko (les bits B0 et B1 de R4 doivent être impérativement à 1). La TGP peut être considérée comme 3 groupes de 256 matrices (0 à 255). L'écran est divisé en 3 zones verticales égales (3x8 lignes). Chaque groupe de la TGP correspond à une zone écran.

Structure de la TNP : La TNP est composée de 768 octets (24x32). A chaque position de la TNP (qui correspond à une position sur l'écran) on peut faire correspondre une matrice de la TGP. Il suffit d'écrire dans l'octet correspondant dans la TNP le numéro de la matrice (0 à 255) dans la TGP. R2 détermine l'adresse de départ de la TNP dans la VRAM par pas de 1 Ko. La TNP doit être considérée comme divisée en 3 zones de 256 octets. Chaque zone correspond à une zone écran (voir TGP). Le contenu d'un octet de la TNP correspond au numéro de la matrice de la TNP DANS LA ZONE CORRESPONDANTE.

Structure de la TC : La TC se compose de 6 Ko. Chaque octet détermine la couleur d'une ligne d'une des matrices de la TGP. Il faut donc 8 octets pour déterminer la couleur d'une matrice. Ainsi, le premier octet de la TC correspond à la première ligne de la première matrice de la TGP. Chaque octet de la TC est divisé en 2 parties : les bits 0 à 3 déterminent la couleur des points à 0 dans la TGP, les bits 4 à 7 déterminent la couleur des points à 1 dans la TGP.

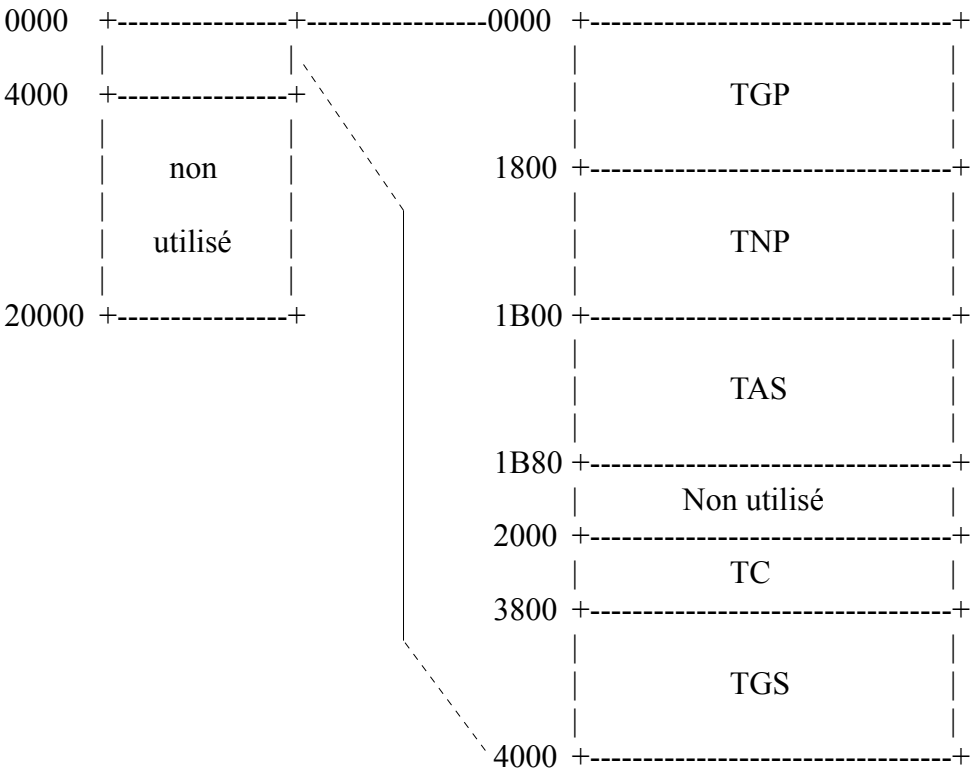
R3 associé à R10 détermine l'adresse de départ de la TC dans la VRAM par pas de 32 octets.

La partie basse du registre R7 est utilisée pour définir la couleur du fond.

Les sprites fonctionnent en mode 1. R5 et R11 permettent de déterminer la position de la TAS. R6 permet de déterminer la position de la TGS.

STRUCTURE DES TABLES EN STANDARD

TGP	0000H - 17FFH	6144 octets
TNP	1800H - 1AFFH	768 octets
TC	2000H - 37FFH	6144 octets
TAS	1B00H - 1B7FH	128 octets
TGS	3800H - 3FFFH	2048 octets
TPL	1B80H - 1B9FH	32 octets



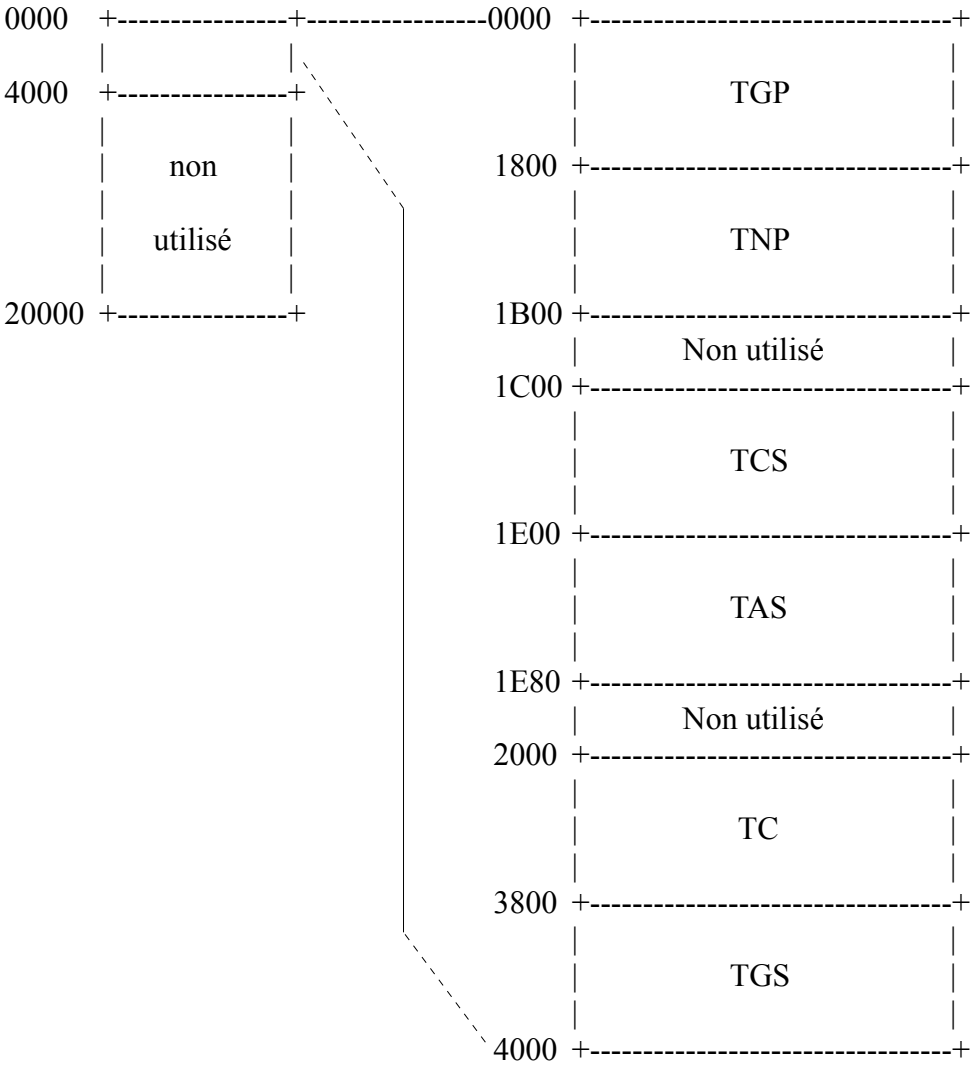
4.6 Mode graphique 3

Le mode graphique 3 est presque identique au mode 2. La seule différence concerne le mode de fonctionnement des sprites (mode 2).

Sélection du mode (R0 & R1) :	M1	M2	M3	M4	M5
	0	0	0	1	0

STRUCTURE DES TABLES EN STANDARD

TGP	0000H - 17FFH	6144 octets
TNP	1800H - 1AFFH	768 octets
TCS	1C00H - 1DFFH	512 octets
TAS	1E00H - 1E7FH	128 octets
TPL	1E80H - 1E9FH	32 octets
TC	2000H - 37FFH	6144 octets
TGS	3800H - 3FFFH	2048 octets



4.7 Mode graphique 4

Caractéristiques

Mode écran BITMAP	
Taille écran :	256 x 212 ou 256 x 192
Couleurs :	16 couleurs parmi 512
Mode sprite :	Mode 2
Taille mémoire d'un écran :	32 Ko

Contrôle

Graphiques :	TNP dans la VRAM
Couleur de fond :	4 bits bas de R7
Sprites :	TAS et TGS dans la VRAM

Sélection du mode (R0 & R1) :	M1	M2	M3	M4	M5
	0	0	1	1	0

Le bit LN (bit 7 de R9) permet de définir la résolution 256 x 192 (0) ou 256 x 212 (1).

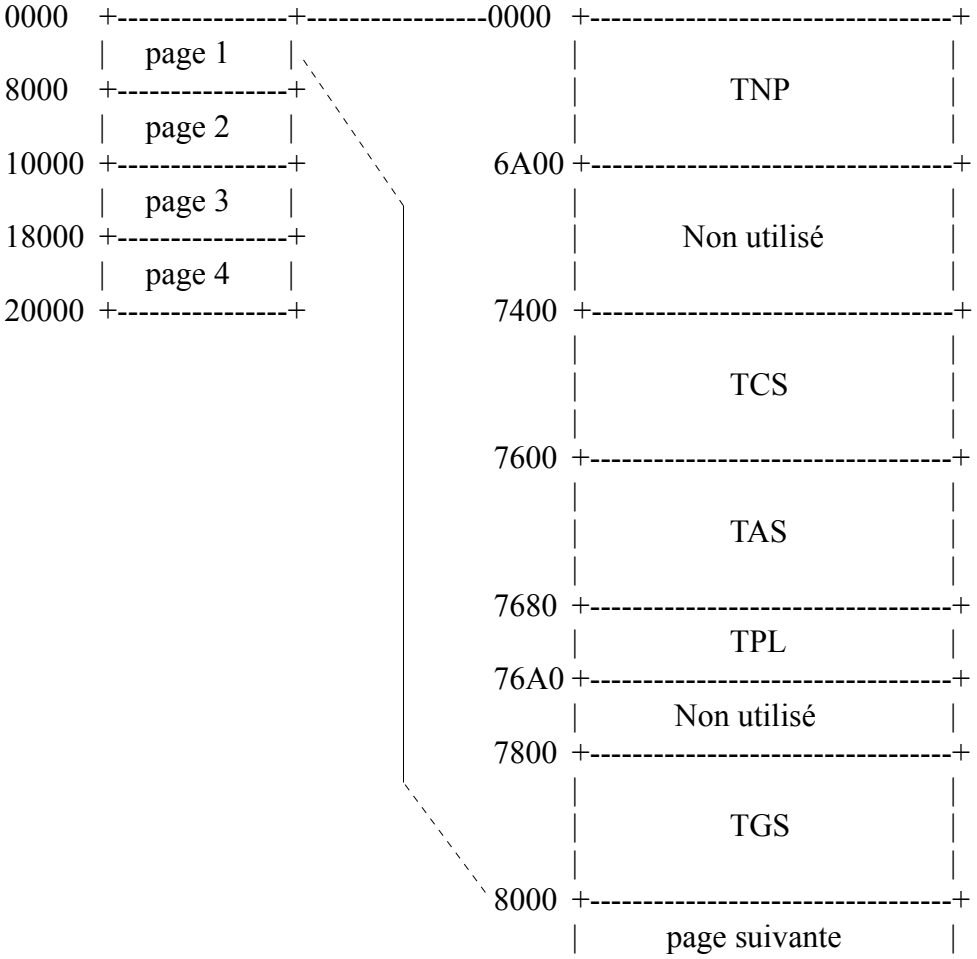
Structure de la TNP : La TNP est constituée de 1 octet pour chaque couple de 2 points consécutifs horizontalement. Il faut donc au maximum 27136 octets (256*212/2) pour mémoriser l'état de chaque point. On dispose donc de 4 bits par point pour déterminer une couleur parmi 16. R2 détermine l'adresse de départ de la TNP par pas de 32 Ko (4 pages possibles). Les bits B0 à B4 de R2 doivent être impérativement à 1.

La partie basse du registre R7 est utilisée pour définir la couleur du fond.

Les sprites fonctionnent en mode 2. R5 et R11 permettent de déterminer la position de la TAS. R6 permet de déterminer la position de la TGS.

STRUCTURE DES TABLES EN STANDARD

TNP	0000H - 69FFH	27136 octets
TCS	7400H - 75FFH	512 octets
TAS	7600H - 767FH	128 octets
TPL	7680H - 769FH	32 octets
TGS	7800H - 7FFFH	2048 octets



4.8 Mode graphique 5

Caractéristiques

Mode écran BITMAP

Taille écran :	512 x 212 ou 512 x 192
Couleurs :	4 couleurs parmi 512
Mode sprite :	Mode 2
Taille mémoire d'un écran :	32 Ko

Contrôle

Graphiques :	TNP dans la VRAM
Couleur de fond :	4 bits bas de R7
Sprites :	TAS et TGS dans la VRAM

Sélection du mode (R0 & R1) :	M1	M2	M3	M4	M5
	0	0	0	0	1

Le bit LN (bit 7 de R9) permet de définir la résolution 512 x 192 (0) ou 512 x 212 (1).

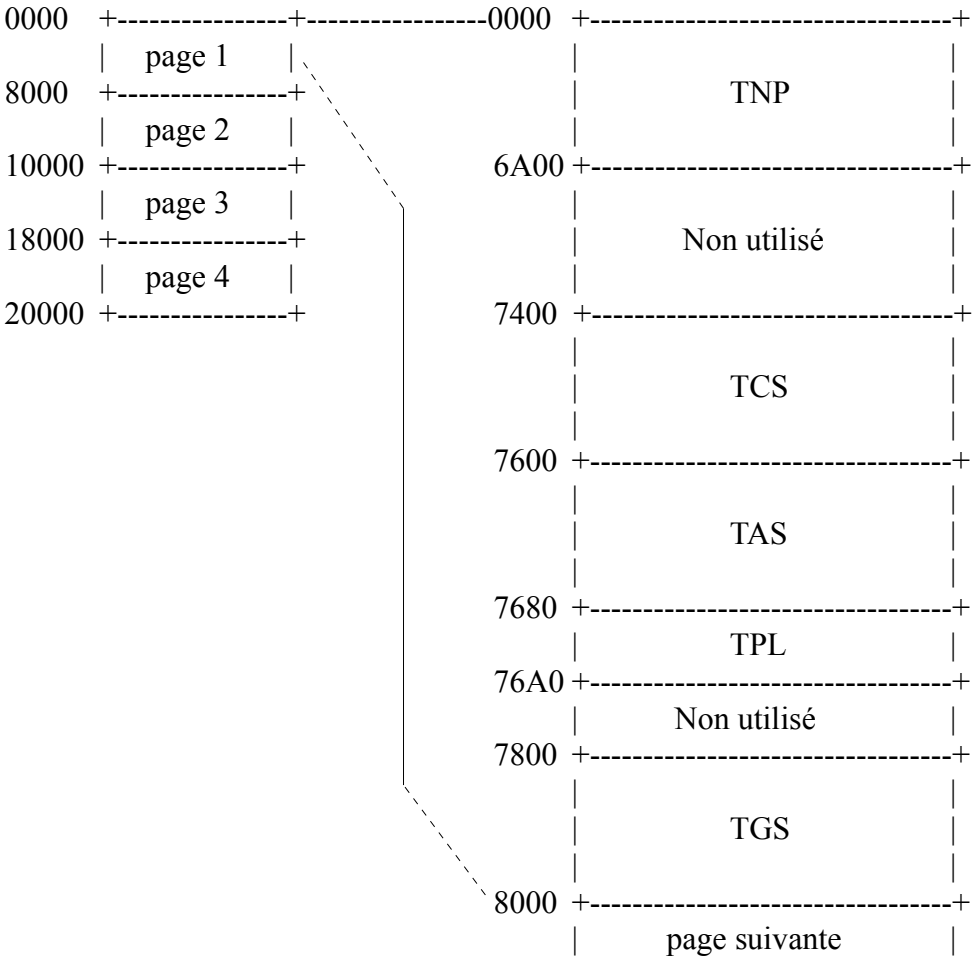
Structure de la TNP : La TNP est constituée de 1 octet pour chaque série de 4 points consécutifs horizontalement. Il faut donc au maximum 27136 octets (512*212/4) pour mémoriser l'état de chaque point. On dispose donc de 2 bits par point pour déterminer une couleur parmi 4. R2 détermine l'adresse de départ de la TNP par pas de 32 Ko (4 pages possibles). Les bits B0 à B4 de R2 doivent être impérativement à 1.

La partie basse du registre R7 est utilisée pour définir la couleur du fond.

Les sprites fonctionnent en mode 2. R5 et R11 permettent de déterminer la position de la TAS. R6 permet de déterminer la position de la TGS.

STRUCTURE DES TABLES EN STANDARD

TNP	0000H - 69FFH	27136 octets
TCS	7400H - 75FFH	512 octets
TAS	7600H - 767FH	128 octets
TPL	7680H - 769FH	32 octets
TGS	7800H - 7FFFH	2048 octets



4.9 Mode graphique 6

Caractéristiques

Mode écran BITMAP

Taille écran :	512 x 212 ou 512 x 192
Couleurs :	16 couleurs parmi 512
Mode sprite :	Mode 2
Taille mémoire d'un écran :	128 Ko (2 écrans)

Contrôle

Graphiques :	TNP dans la VRAM
Couleur de fond :	4 bits bas de R7
Sprites :	TAS et TGS dans la VRAM

Sélection du mode (R0 & R1) :	M1	M2	M3	M4	M5
	0	0	1	0	1

Le bit LN (bit 7 de R9) permet de définir la résolution 512 x 192 (0) ou 512 x 212 (1).

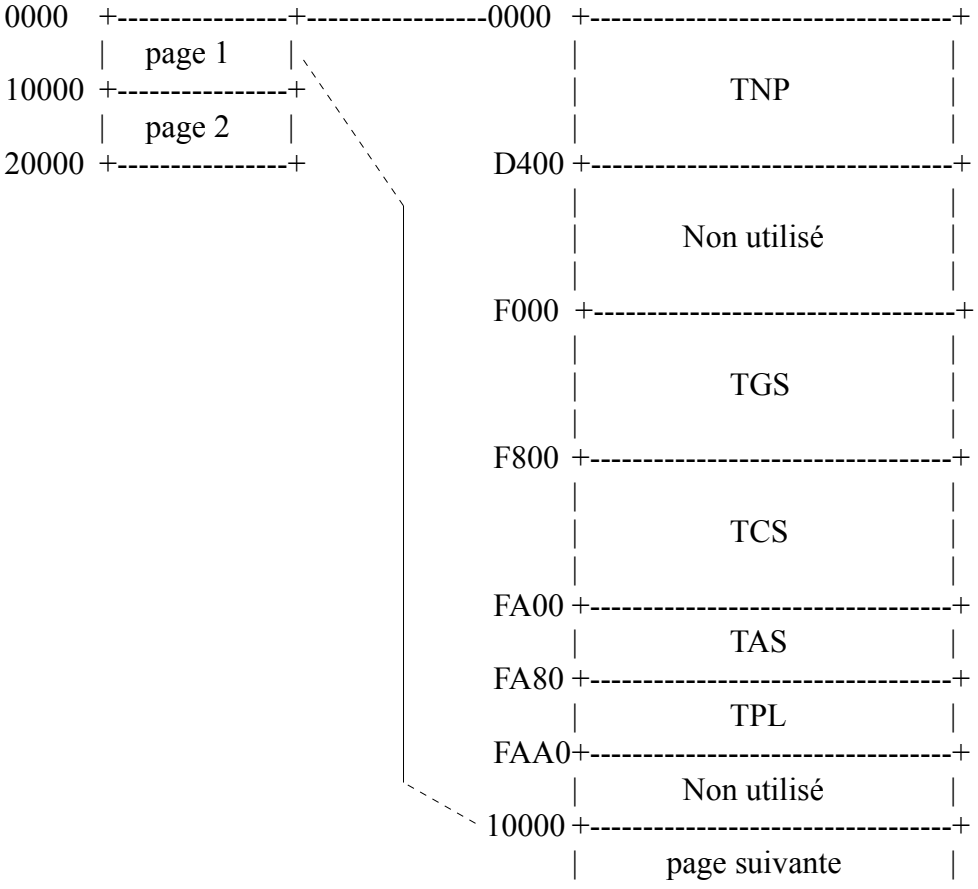
Structure de la TNP : La TNP est constituée de 1 octet pour chaque couple de 2 points consécutifs horizontalement. Il faut donc au maximum 54272 octets (512*212/2) pour mémoriser l'état de chaque point. On dispose donc de 4 bits par point pour déterminer une couleur parmi 16. R2 détermine l'adresse de départ de la TNP par pas de 64 Ko (2 pages possibles). Les bits B0 à B5 de R2 doivent être impérativement à 1.

La partie basse du registre R7 est utilisée pour définir la couleur du fond.

Les sprites fonctionnent en mode 2. R5 et R11 permettent de déterminer la position de la TAS. R6 permet de déterminer la position de la TGS.

STRUCTURE DES TABLES EN STANDARD

TNP	0000H - D3FFH	54272 octets
TGS	F000H - F7FFH	2048 octets
TCS	F800H - F9FFH	512 octets
TAS	FA00H - FA7H	128 octets
TPL	FA80H - FA9FH	32 octets



4.10 Mode graphique 7

Caractéristiques

Mode écran BITMAP

Taille écran :	256 x 212 ou 256 x 192
Couleurs :	256 couleurs
Mode sprite :	Mode 2
Taille mémoire d'un écran :	128 Ko (2 écrans)

Contrôle

Graphiques :	TNP dans la VRAM
Couleur de fond :	4 bits bas de R7
Sprites :	TAS et TGS dans la VRAM

Sélection du mode (R0 & R1) :	M1	M2	M3	M4	M5
	0	0	1	1	1

Le bit LN (bit 7 de R9) permet de définir la résolution 512 x 192 (0) ou 512 x 212 (1).

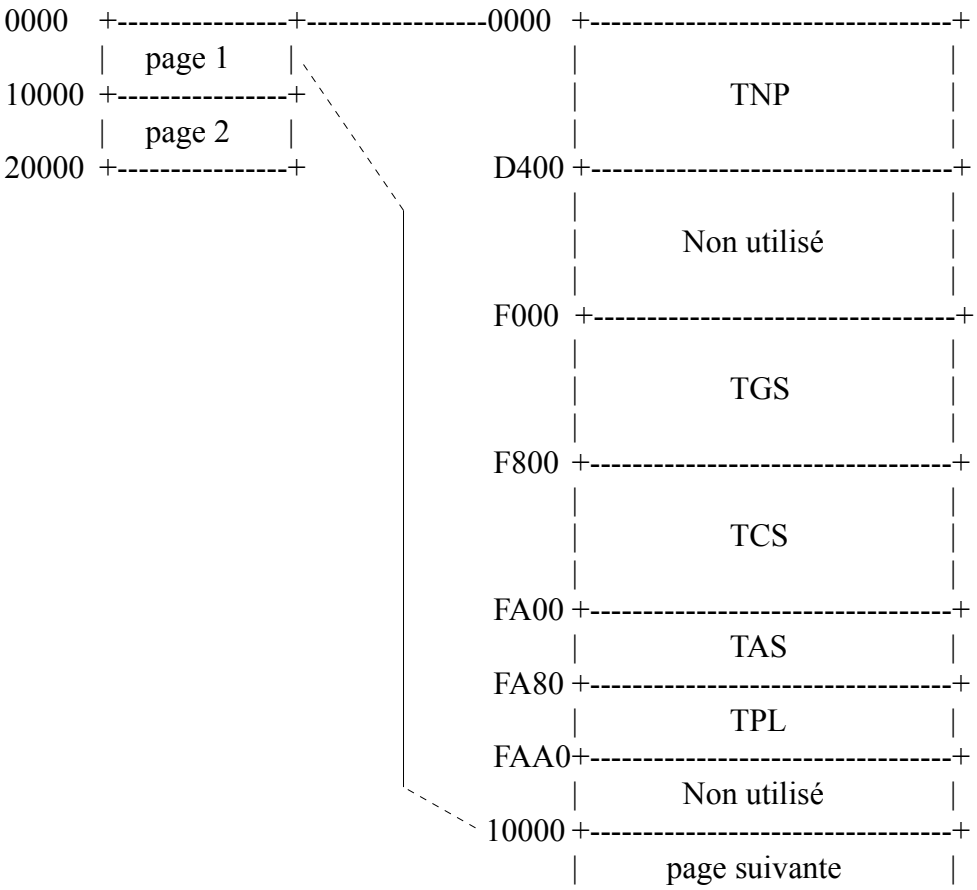
Structure de la TNP : La TNP est constituée de 1 octet pour chaque point. Il faut donc au maximum 54272 octets (256*212) pour mémoriser l'état de chaque point. On dispose donc de 8 bits par point pour déterminer une couleur parmi 256. R2 détermine l'adresse de départ de la TNP par pas de 64 Ko (2 pages possibles). Les bits B0 à B5 de R2 doivent être impérativement à 1.

La partie basse du registre R7 est utilisée pour définir la couleur du fond.

Les sprites fonctionnent en mode 2. R5 et R11 permettent de déterminer la position de la TAS. R6 permet de déterminer la position de la TGS.

STRUCTURE DES TABLES EN STANDARD

TNP	0000H - D3FFH	54272 octets
TGS	F000H - F7FFH	2048 octets
TCS	F800H - F9FFH	512 octets
TAS	FA00H - FA7FH	128 octets
TPL	FA80H - FA9FH	32 octets



4.11 Adressage et utilisation des tables

Il est évident que chaque table possède en standard une adresse de base définie par le système.

Analysons la structure de ces tables et leurs adresses pour les modes courants du système, autrement dit le mode texte, le mode graphique 1 et le mode graphique 2.

Adressage en mode texte :

En mode SCREEN 0 (TEXTE), les adresses comprises entre 0 et 959 de la vidéoram (en hexadécimal 0000 à 03BF) contiennent les codes des caractères affichés en position correspondante sur l'écran. Autrement dit, la TNP est positionnée en 0.

La TGP quant à elle est positionnée en 2048 (en hexadécimal 0800), ce qui veut dire que les adresses comprises entre 2048 et 4095 (en hexadécimal 0800 et 0FFF) sont utilisées pour le dessin des caractères, celui correspondant à la valeur 0 se trouvant codé sur les octets compris entre 2048 et 2055 (8 octets).

La TNP et la TGP sont seules utilisées en mode TEXTE.

Exemple : Nous voulons afficher un caractère dont voici le dessin (1 représente un point allumé et 0 un point éteint et ce, au milieu de l'écran). Ce caractère doit remplacer la lettre A.

DESSIN (matrice de 6 x 8)

```
11111100        ******
10110100        *  **  *
10110100        *  **  *
11111100        ******
11111100        ******
10110100        *  **  *
10110100        *  **  *
11111100        ******
```

On calcule la valeur des 8 octets (1 par ligne) :

ligne 1	11111100 = FCH = 252
ligne 2	10110100 = B4H = 180
ligne 3	10110100 = B4H = 180
ligne 4	11111100 = FCH = 252
ligne 5	11111100 = FCH = 252
ligne 6	10110100 = B4H = 180
ligne 7	10110100 = B4H = 180
ligne 8	11111100 = FCH = 252

On détermine la valeur de la lettre A en prenant son code ASCII.

A = 41H = 65.

On détermine m'adresse du milieu d'écran (environ 460). On écrit la valeur du code de A au milue de l'écran :

```
10 CLS
20 VPOKE 460, 65
```

On détermine la position de A dans la TGP (facile, c'est 2048 + 8 x 65).

Soit AD cette adresse. Il ne reste plus qu'à inscrire les valeurs de chaque ligne dans les 8 adresses successives :

```
30 AD = 2568
40 VPOKE AD, 252
50 VPOKE AD+1, 180
50 VPOKE AD+2, 180
50 VPOKE AD+3, 252
50 VPOKE AD+4, 252
50 VPOKE AD+5, 180
50 VPOKE AD+6, 180
50 VPOKE AD+7, 252
```

Et le tour est joué !

Remarque : Bien sûr les instructions vues ci-dessus doivent se dérouler dans un programme de façon successive. A partir de cet instant, la lettre A de votre clavier est remplacée par le petit dessin.

Listez le programme et regardez le A de AD !

Adressage en mode graphique 1 :

Dans ce mode, toutes les tables sont actives.

La TGP, qui contient les dessins des 256 caractères en format 8x8 est localisée de 0 à 6143 (0000H à 17FFH).

La TC commence en 8192 (2000H) et occupe 32 octets.

La TNP est positionnée à l'adresse 1800H.

La TGS est positionnée à l'adresse 3800H.

La TAS est positionnée à l'adresse 1B00H.

Adressage en mode graphique 2 :

Dans ce mode (SCREEN 2), toutes les tables sont actives.

La TGP, qui contient l'information sur les points allumés ou éteints se trouve de 0 à 6143 (0000H à 17FFH).

L'octet 0 contient l'information sur les points de coordonnées 0, 0 à 7,0 (format colonne, ligne).

L'octet 1 contient l'information sur les points de coordonnées 0, 1 à 7,1

L'octet 7 contient l'information sur les points de coordonnées 0, 7 à 7,7

L'octet 8 contient l'information sur les points de coordonnées 8, 0 à 15,0

Et ainsi de suite.

Pour allumer le point de coordonnées X, Y avec 0 < X < 255 et 0 < Y < 191, il suffit de faire :

VPOKE AD,(2(7-(XMOD8)))OR VPEEK AD

avec AD=X-XMOD8 + YMOD8+256X(Y\8)

La TC commence en 8192 et termine en 14335 (hexadécimal 2000H à 37FFH).

La mémoire est organisée comme suit : chaque adresse contient la couleur d'un groupe de huit points horizontaux.

L'octet d'adresse 8192 s'occupe des 8 points de coordonnées 0,0 à 0,7. Le suivant (8193) s'occupe des 8 points de coordonnées 8, 0 à 15,0 etc.

Exemple : pour afficher le point de coordonnées X,Y de la couleur CL avec les points éteints de couleur CF :

VPOKE AD,V

avec AD=Y+320+X\8+8192 et V=16-CL+CF.

La TNP est positionnée à partir de l'adresse 1800H et occupe 768 octets.

La TGS est positionnée en 3800H.

La TAS est positionnée en 1B00H.

RESUME : TABLE DES ADRESSES DES TABLES

ECRAN	TNP	TGP	TC	TGS	TCS	TAS	Pal	LG	NPG
T1 (40)	0000	0800	-	-	-	-	0400	1000	32
T2 (80)	0000	1000	0800	-	-	-	0F00	2000	16
G1	1800	0000	2000	3800	-	1B00	2020	4000	8
G2	1800	0000	2000	3800	-	1B00	2020	4000	8
MULTI	0800	0000	-	3800	-	1B00	2020	4000	8
G3	1800	0000	2000	3800	1C00	1E00	2020	4000	8
G4	0000	-	-	7800	7400	7600	7680	8000	4
G5	0000	-	-	7800	7400	7600	7680	8000	4
G6	0000	-	-	F000	F800	FA00	FA80	10000	2
G7	0000	-	-	F000	F800	FA00	FA80	10000	2

5 Les commandes du VDP

5.1 Généralités

Les commandes évoluées du VDP permettent de réaliser facilement les fonctions graphiques élémentaires. Ces commandes simplifient grandement l'écriture des routines internes du Bios.

Le type de commande est déterminé par le contenu des 4 bits hauts du registre R46.

Les paramètres de la commande sont déterminés par les registres R32 à R45. La commande est exécutée par le positionnement de R46. La fin de la commande peut être testée par l'intermédiaire du bit 0(CE) du registre d'état S2.

Les commandes sont valables en mode graphique 4 à graphique 7.

Les commandes utilisent un adressage direct en coordonnées X, Y. C'est le VDP lui-même qui réalise la conversion entre les coordonnées et les adresses de la vidéoram.

La bonne adresse mémoire est utilisée en fonction des coordonnées X, Y ainsi que de la page courante (R2).

Lors de l'exécution d'une commande LINE, PSET ou d'un transfert logique (LOGICAL MOVE), les 4 bits bas de R46 permettent de déterminer l'opération logique à exécuter entre les couleurs des points présents sur l'écran et de ceux amenés par la commande.

Remarque : Le temps d'exécution des commandes est sensiblement amélioré si le bit 1 de R8 (SPD) est mis à 1. Ce bit inhibe l'affichage des sprites. De même le positionnement à 0 du bit 6 de R1 (BL) qui inhibe l'affichage de l'écran améliore la vitesse de traitement d'une commande.

5.2 HMMC

Cette commande transfère des données en provenance de l'unité centrale vers la mémoire RAM du VDP ou vers l'extension mémoire.

Processus :

1 - Sélectionner le type de mémoire (VRAM ou EXPANSION RAM) par l'intermédiaire de MXD (R45).

2 - Positionner correctement les registres DX et DY (R36/37 et R38/39). Ils doivent contenir les coordonnées X (0 à 511) et Y (0 à 1023) d'un coin de rectangle contenant la zone de réception en VRAM.

3 - Positionner correctement les registres NX et NY (R40/41 et R42/43). Ils doivent contenir le nombre de points à transférer dans l'axe des X (0 à 511) et dans l'axe des Y (0 à 1023).

4 - Positionner les indicateurs de directions DIX et DIY (voir R45).

5 - Mettre dans CLR (R44) le premier octet à transférer.

6 - Ecrire 0F0H dans CMR (R46) pour enclencher le processus.

7 - Tester le bit CE de S2. Ce bit passe à 0 quand la commande est terminée. Si le bit est à 1, passer au point 8. Si le bit est à 0, la commande est terminée.

8 - Tester le bit TR de S2. Ce bit devient 1 lorsque l'octet est transmis. Si le bit est à 0, recommencer le point 8. Quand le bit passe à 1, passer au point 9.

9 - Charger l'octet suivant dans R44 et retourner au point 7.

Remarque : En mode G7 (graphique 7), un octet CPU correspond à un point écran (VRAM). En mode G4 et G6, un octet CPU correspond à 2 points écrans. En mode G5, un octet CPU correspond à 4 points.

5.3 YMMM

Cette commande permet de transférer rapidement un bloc d'information de la VRAM (ou de la RAM EXPANSION) vers la VRAM (ou la RAM EXPANSION).

Cette commande déplace le bloc de données sur l'axe Y uniquement. La coordonnée X du point de départ est identique à celle du point d'arrivée.

Processus :

1 - Sélectionner le type de mémoire (VRAM ou EXPANSION RAM) par l'intermédiaire de MXD (R45).

2 - Positionner correctement le registre DY (R38/39). Il doit contenir la coordonnée Y du point d'arrivée (0 à 1023) d'un coin de rectangle à déplacer en VRAM.

3 - Positionner correctement les registres DX et SY (R36/37 et R34/35). Ils doivent contenir les coordonnées X (0 à 511) et Y (0 à 1023) du point de départ d'un coin de rectangle à déplacer en VRAM.

4 - Positionner correctement le registre NY (R42/43). Il doit contenir le nombre de points à transférer dans l'axe des Y (0 à 1023).

5 - Positionner les indicateurs de direction DIX et DIY (voir R45).

6 - Ecrire 0E0H dans CMR (R46) pour enclencher le processus.

7 - Tester le bit CE de S2. Ce bit passe à 0 quand la commande est terminée.

5.4 HMMM

Cette commande permet de transférer rapidement un bloc d'information de la VRAM (ou de la RAM EXPANSION) vers la VRAM (ou la RAM EXPANSION).

Processus :

1 - Sélectionner le type de mémoire (VRAM ou EXPANSION RAM) par l'intermédiaire de MXD (R45).

2 - Positionner correctement les registres SX et SY (R32/33 et R34/35). Ils doivent contenir les coordonnées X et Y du point de départ d'un coin de rectangle à déplacer en VRAM.

3 - Positionner correctement les registres DX et DY (R36/37 et R38/39). Ils doivent contenir les coordonnées X (0 à 511) et Y (0 à 1023) du point d'arrivée d'un coin de rectangle à déplacer en VRAM.

4 - Positionner correctement les registres NX et NY (R40/41 et R42/43). Ils doivent contenir le nombre de points à transférer dans l'axe des X (0 à 511) et dans l'axe des Y (0 à 1023).

5 - Positionner les indicateurs de direction DIX et DIY (voir R45).

6 - Ecrire 0D0H dans CMR (R46) pour enclencher le processus.

7 - Tester le bit CE de S2. Ce bit passe à 0 quand la commande est terminée.

5.5 HMMV

Cette commande permet de colorier (PAINT) une zone rectangulaire de la VRAM (ou de la RAM EXPANSION).

Processus :

1 - Sélectionner le type de mémoire (VRAM ou EXPANSION RAM) par l'intermédiaire de MXD (R45).

2 - Positionner correctement les registres DX et DY (R36/37 et R38/39). Ils doivent contenir les coordonnées X (0 à 511) et Y (0 à 1023) d'un coin de rectangle à colorier en VRAM.

3 - Positionner correctement les registres NX et NY (R40/41 et R42/43). Ils doivent contenir le nombre de points à colorier dans l'axe des X (0 à 511) et dans l'axe des Y (0 à 1023).

4 - Positionner les indicateurs de direction DIX et DIY (voir R45).

5 - Mettre la couleur dans CLR (R44).

6 - Ecrire 0C0H dans CMR (R46) pour enclencher le processus.

7 - Tester le bit CE de S2. Ce bit passe à 0 quand la commande est terminée.

5.6 LMMC

Cette commande transfère des données en provenance de l'unité centrale vers la mémoire RAM du VDP ou vers l'extension mémoire. Une opération logique déterminée par le contenu du registre de commande est exécutée entre chaque point source et chaque point arrivée.

Processus :

1 - Sélectionner le type de mémoire (VRAM ou EXPANSION RAM) par l'intermédiaire de MXD (R45).

2 - Positionner correctement les registres DX et DY (R36/37 et R38/39). Ils doivent contenir les coordonnées X (0 à 511) et Y (0 à 1023) du point d'arrivée d'un coin de rectangle contenant la zone de destination en VRAM.

3 - Positionner correctement les registres NX et NY (R40/41 et R42/43). Ils doivent contenir le nombre de points à transférer dans l'axe des X (0 à 511) et dans l'axe des Y (0 à 1023).

4 - Positionner les indicateurs de direction DIX et DIY (voir R45).

5 - Mettre dans CLR (R44) le premier octet à transférer.

6 - Ecrire 0BXH dans CMR (R46) pour enclencher le processus. X contient le code de l'opération logique à effectuer (voir description du registre R46).

7 - Tester le bit CE de S2. Ce bit passe à 0 quand la commande est terminée. Si le bit est à 1, passer au point 8. Si le bit est à 0, la commande est terminée.

8 - Tester le bit TR de S2. Ce bit devient 1 lorsque l'octet est transmis. Si le bit est à 0, recommencer le point 8. Quand le bit passe à 1, passer au point 9.

9 - Charger l'octet suivant dans R44 et retourner au point 7.

Remarque : En mode G7 (graphique 7), un octet CPU correspond à un point écran (VRAM). En mode G4 et G6, un octet CPU correspond à 2 points écrans. En mode G5, un octet CPU correspond à 4 points.

5.7 LMCM

Cette commande transfère des données en provenance de la mémoire RAM vers l'unité centrale.

Une opération logique déterminée par le contenu du registre de commande est exécutée entre chque point source et chaque point arrivée.

Processus :

1 - Sélectionner le type de mémoire (VRAM ou EXPANSION RAM) par l'intermédiaire de MXD (R45).

2 - Positionner correctement les registres SX et SY (R32/33 et R34/35). Ils doivent contenir les coordonnées X et Y du point de départ d'un coin de rectangle contenant la zone source en VRAM.

3 - Positionner correctement les registres NX et NY (R40/41 et R42/43). Ils doivent contenir le nombre de points à transférer dans l'axe des X (0 à 511) et dans l'axe des Y (0 à 1023).

4 - Positionner les indicateurs de direction DIX et DIY (voir R45).

5 - Lire S2 simplement pour le remettre à 0.

6 - Ecrire 0AXH dans CMR (R46) pour enclencher le processus. X contient le code de l'opération logique à effectuer (voir description du registre R46).

7 - Tester le bit TR de S2. Ce bit devient 1 lorsque l'octet est transmis. Si le bit est à 0, aller au point 9 sinon aller au point 8.

8 - Lire le registre d'état S7 qui contient l'octet transféré.

9 - Tester le bit CE de S2. Ce bit passe à 0 quand la commande est terminée. Si le bit est à 1, retourner au point 7. Si le bit est à 0, la commande est terminée.

Remarque : En mode G7 (graphique 7), un octet CPU correspond à un point écran (VRAM). En mode G4 et G6, un octet CPU correspond à 2 points écrans. En mode G5, un octet CPU correspond

à 4 points.

5.8 LMMM

Cette commande permet de transférer logiquement un bloc d'information de la VRAM (ou de la RAM EXPANSION) vers la VRAM (ou la RAM EXPANSION)

Une opération logique déterminée par le contenu du registre de commande est exécutée entre chque point source et chaque point arrivée.

Processus :

1 - Sélectionner le type de mémoire (VRAM ou EXPANSION RAM) par l'intermédiaire de MXD (R45).

2 - Positionner correctement les registres SX et SY (R32/33 et R34/35). Ils doivent contenir les coordonnées X et Y du point de départ d'un coin de rectangle à déplacer en VRAM.

3 - Positionner correctement les registres DX et DY (R36/37 et R38/39). Ils doivent contenir les coordonnées X (0 à 511) et Y (0 à 1023) du point d'arrivée d'un coin du rectangle à déplacer en VRAM.

4 - Positionner correctement les registres NX et NY (R40/41 et R42/43). Ils doivent contenir le nombre de points à transférer dans l'axe des X (0 à 511) et dans l'axe des Y (0 à 1023).

5 - Positionner les indicateurs de direction DIX et DIY (voir R45).

6 - Ecrire 09XH dans CMR (R46) pour enclencher le processus. X contient le code de l'opération logique à effectuer (voir description du registre R46).

7 - Tester le bit CE de S2. Ce bit passe à 0 quand la commande est terminée.

5.9 LMMV

Cette commande permet de colorier (PAINT) une zone rectangulaire de la VRAM (ou de la RAM EXPANSION).

Une opération logique déterminée par le contenu du registre de commande est exécutée entre chque point source et chaque point arrivée.

Processus :

1 - Sélectionner le type de mémoire (VRAM ou EXPANSION RAM) par l'intermédiaire de MXD (R45).

2 - Positionner correctement les registres DX et DY (R36/37 et R38/39). Ils doivent contenir les coordonnées X (0 à 511) et Y (0 à 1023) d'un coin de rectangle à colorier en VRAM.

3 - Positionner correctement les registres NX et NY (R40/41 et R42/43). Ils doivent contenir le nombre de points à colorier dans l'axe des X (0 à 511) et dans l'axe des Y (0 à 1023).

4 - Positionner les indicateurs de direction DIX et DIY (voir R45).

5 - Mettre la couleur dans CLR (R44).

6 - Ecrire 08XH dans CMR (R46) pour enclencher le processus. X contient le code de l'opération logique à effectuer (voir description du registre R46).

7 - Tester le bit CE de S2. Ce bit passe à 0 quand la commande est terminée.

5.10 LINE

La commande LINE trace un segment de droite qui correspond à l'hypoténuse d'un triangle rectangle dont les côtés de l'angle droit sont parallèles aux axes X et Y. On donne la taille des deux côtés de l'angle droit, le sens du tracé, la coordonnée du point d'un des angles opposés à l'angle droit et la direction du plus grand côté.

Cette définition peut paraître difficile puisque les coordonnées X, Y de 2 points permettent de définir de façon univoque un segment de droite. C'est la structure de condition matérielle du VDP qui impose cette méthodologie.

Une opération logique déterminée par le contenu du registre de commande est exécutée entre chque point source et chaque point arrivée.

Processus :

1 - Sélectionner le type de mémoire (VRAM ou EXPANSION RAM) par l'intermédiaire de MXD (R45).

2 - Positionner correctement les registres DX et DY (R36/37 et R38/39). Ils doivent contenir les coordonnées X (0 à 511) et Y (0 à 1023) d'une des extrémités du segment (un des angles opposés à l'angle droit).

3 - DX doit contenir la taille (en points) du plus grand des deux côtés de l'angle droit. DY doit contenir la taille (en points) du plus petit des deux côtés de l'angle droit. Si les deux côtés sont de même taille, DX et DY contiennent la même valeur.

4 - Le bit MAJ (bit 0 de R45) contient 0 si le plus grand côté du triangle est parallèle à l'axe des X et 1 si le plus grand côté du triangle est parallèle à l'axe des Y. Si les deux côtés sont de même taille, ce bit est sans importance.

5 - Positionner les indicateurs de direction DIX et DIY (voir R45).

6 - CLR doit contenir la couleur du segment.

7 - Ecrire 07XH dans CMR (R46) pour enclencher le processus. X contient le code de l'opération logique à effectuer (voir description du registre R46).

8 - Tester le bit CE de S2. Ce bit passe à 0 quand la commande est terminée.

5.11 SRCH

La commande SRCH recherche une couleur de bord dans la VRAM en partant d'un point spécifié et en se dirigeant à gauche ou à droite de ce point. A l'issue de la commande, les coordonnées du point trouvé figure dans des registres.

Processus :

1 - Sélectionner le type de mémoire (VRAM ou EXPANSION RAM) par l'intermédiaire de MXD (R45).

2 - Positionner correctement les registres SX et SY (R32/33 et R34/35). Ils doivent contenir les coordonnées X et Y du point de départ de la recherche.

3 - Positionner l'indicateur de direction DIX dans le sens de la recherche (0 à droite, 1 à gauche).

4 - Positionner EQ (bit 1 de R45) en fonction du type de recherche (1 = arrêt sur couleur trouvée, 0 = arrêt sur couleur différente).

5 - CLR doit contenir la couleur à rechercher.

6 - Ecrire 060H dans CMR (R46) pour enclencher le processus.

7 - Tester le bit CE de S2. Ce bit passe à 0 quand la commande est terminée.

8 - Le bit BD (bit 4 de S2) passe à 1 si la couleur désirée est trouvée (si EQ = 1) ou si une autre couleur que celle spécifiée est trouvée (EQ = 0). A ce moment, il faut lire les registres d'état S8 et S9 qui contiennent la coordonnée X du point d'arrêt de la recherche.

5.12 PSET

La commande PSET permet de positionner un point dans une couleur quelconque à une position quelconque de la vidéoram.

Une opération logique déterminée par le contenu du registre de commande est exécutée entre chque point source et chaque point arrivée.

Processus :

1 - Sélectionner le type de mémoire (VRAM ou EXPANSION RAM) par l'intermédiaire de MXD (R45).

2 - Positionner correctement les registres DX et DY (R36/37 et R38/39). Ils doivent contenir les coordonnées X (0 à 511) et Y (0 à 1023) du point à allumer.

3 - CLR doit contenir la couleur du point à allumer.

4 - Ecrire 05XH dans CMR (R46) pour enclencher le processus. X contient le code de l'opération logique à effectuer (voir description du registre R46).

5 - Tester le bit CE de S2. Ce bit passe à 0 quand la commande est terminée.

5.13 POINT

La commande POINT permet de tester la couleur courante d'un point situé à une position quelconque de la vidéoram.

Processus :

1 - Sélectionner le type de mémoire (VRAM ou EXPANSION RAM) par l'intermédiaire de MXD (R45).

2 - Positionner correctement les registres SX et SY (R32/33 et R34/35). Ils doivent contenir les coordonnées X et Y du point à tester.

3 - Ecrire 040H dans CMR (R46) pour enclencher le processus.

4 - Tester le bit CE de S2. Ce bit passe à 0 quand la commande est terminée.

5 - A l'issue de la commande, S7 contient la couleur du point testé.

5.14 Etat des registres après exécution

Commande	SX	SY	DX	DY	NX	NY	CLR	CMRH	CMRL	ARG
HMMC	=	=	=	*	=	#	=	0	=	=
YMMM	=	*	=	*	=	#	=	0	=	=
HMMM	=	*	=	*	=	#	=	0	=	=
HMMV	=	=	=	*	=	#	=	0	=	=
LMMC	=	=	=	*	=	#	=	0	=	=
LMCM	=	*	=	=	=	#	$	0	=	=
LMMM	=	*	=	*	=	#	=	0	=	=
LMMV	=	=	=	*	=	#	=	0	=	=
LINE	=	=	=	*	=	#	=	0	=	=
SRCH	=	=	=	=	=	=	=	0	=	=
PSET	=	=	=	=	=	=	=	0	=	=
POINT	=	=	=	=	=	=	$	0	=	=

Signification des symboles :

= : non modifié

0 : vaut 0 (les 4 bits de poids fort de CMR)

* : vaut la coordonnée du point à l'issue de la commande

$: code de couleur trouvé

: valeur du compteur NYB lors de la détection de fin d'écran

6 Les sprites

6.1 Généralités

Le VDP 9938 peut gérer 32 sprites. La taille d'un sprite est de 8x8 ou 16x16 points. La taille horizontale d'un point d'un sprite est toujours 1/256 de la taille totale de l'écran. Autrement dit, dans les modes graphiques avec une résolution de 512 points, un point de sprite recouvre 2 points de l'image.

Les sprites fonctionnent selon deux modes différents. Le mode correct est choisi automatiquement par le mode graphique.

- le mode 1 : en mode graphique 1, 2 et Multicolore
- le mode 2 : en mode graphique 3, 4, 5, 6 et 7

Caractéristiques particulières du mode 1 :

- Sur une même ligne horizontale, on peut afficher 4 sprites.
- On peut obtenir le numéro du cinquième sprite sur la même horizontale.
- La collision de deux sprites est détectée par recouvrement de parties des matrices de sprites dont les bits sont à 1.

- Les couleurs sont limitées à 2.

Caractéristiques du mode 2 :

- Sur une même ligne horizontale, on peut afficher 8 sprites.
- On peut obtenir le numéro du neuvième sprite sur la même horizontale.
- La collision de deux sprites est détectée par recouvrement de partie des matrices de sprites solides (non transparents).
- Lors d'une collision, les coordonnées de la collision se retrouvent dans les registres S3 et S5.
- La gestion de priorité peut être modifiée.

6.2 Les sprites en mode 1

Le mode 1 (ancien mode du MSX1) permet de gérer 32 sprites (0 à 31).

Le VDP peut afficher 32 sprites sur ses 32 plans les plus prioritaires, à raison d'un sprite par plan.

La structure des tables qui définissent les sprites facilite l'animation de ces derniers.

La localisation d'un sprite sur l'écran est définie par le point supérieur gauche de son patron. Ainsi un sprite de 8x8 points affiché en 128, 96 occupera les points de 128, 96 à 137, 104.

Le sprite peut être déplacé simplement en changeant son origine, autrement dit en changeant les coordonnées de son coin supérieur gauche.

Ce système permet une grande simplicité et une grande rapidité dans la programmation d'objets en mouvements.

Le nombre de sprites sur une même horizontale est limité à 4. La détection de présence d'un 5ème sprite est réalisée par le bit 6 du registre de statut S0. Les bits B0 à B4 donnent alors le numéro de ce 5ème sprite.

Lors d'une détection de collision, le bit 5 du registre de statut S0 est mis à 1.

La taille des sprites est gérée par le bit 1 (SI) sur registre R1.

SI = 0 : 8x8 points

SI = 1 : 16x16 points

Le bit 0 de R1 (MAG) permet de définir un facteur multiplicatif pour la taille des sprites.

MAG = 0 : Taille normale

MAG = 1 : Taille double

Voici un tableau récapitulatif de la taille des sprites en fonction des bits B0 et B1 du registre 1 :

B1 taille	B0 agrandissement	dimensions	résolution	Nb d'octets dans la TGS
0	0	8x8	1 point	8
1	0	16x16	1 point	32
0	1	16x16	2x2 points	8
1	1	32x32	2x2 points	32

Rappel : En Basic, la taille et le facteur d'agrandissement sont commandés après le 2ème paramètre de l'instruction SCREEN :

SCREEN 2,0 : mode graphique 2, taille 8x8

SCREEN 2,1 : mode graphique 2, mais avec agrandissement

SCREEN 2,2 : mode graphique 2, taille 16x16

SCREEN 2,3 : idem 2,2 mais avec agrandissement

Les sprites sont définis par deux tables : la TAS et la TGS.

La TAS

La TAS spécifie les coordonnées, le numéro et la couleur du sprite. La TAS comporte 4 octets par sprite.

Octet 1 : position verticale du point supérieur gauche. Si la valeur de cet octet est égale à 208, tous les sprites de priorité inférieure (numéro supérieur) ne seront pas affichés.

Octet 2 : position horizontale du coin supérieur gauche.

Octet 3 : pointeur relatif à la TGS. Il contient le numéro de matrice dans la TGS. En mode 16x16 il y a 4 matrices par sprite. Vous pouvez alors spécifier le numéro de la matrice de votre choix.

Octet 4 : Les 4 bits les moins significatifs indiquent la couleur des bits à 1 dans le patron défini dans la TGS (les bits à 0 sont transparents). Le bit B7 permet de décaler le sprite de 32 points vers la gauche. Ce bit permet de faire des apparitions partielles de sprites en partant du bord gauche de l'écran (voir déplacement des sprites). Les bits B6, B5 et B4 sont inutilisés et doivent valoir 0.

Les 4 premiers octets de la TAS sont relatifs au sprite du plan 0, les 4 suivants au sprite du plan 1 et ainsi de suite jusqu'au plan 31. Il y a donc 128 octets dans la TAS.

Remarque : Les 2 premiers octets définissent les coordonnées du sprite par rapport au coin supérieur gauche de l'écran. La valeur de la coordonnée verticale du point supérieur gauche a été fixée à -1 et la valeur de la coordonnées horizontale de ce coin a été fixée à 0.

Pour afficher un sprite dans le coin supérieur gauche de l'écran, il faut donc mettre le premier octet relatif au plan dans la TAS à -1 et le second à 0.

LA TGS

La TGS est une table de 2048 octets organisée en 256 blocs de 8 octets. Elle définit le patron du sprite. La position de départ dans la TGS dans la VRAM est déterminée par le registre R6.

Le troisième octet de chaque entrée dans la TAS spécifie le bloc correspondant dans la TGS.

La TGS est organisée comme suit : les blocs de 8x8 octets sont groupés 4 par 4. Si le sprite est au format 8x8 ou 16x16 avec le bit d'agrandissement = 1, un seul bloc de 8 octets est occupé sur les 4, les 3 autres n'étant pas considérés par le VDP. Si le sprite est au format 16x16 avec le bit d'agrandissement à 0 ou au format 32x32, les 4 blocs sont visualisés dans l'ordre suivant :

0	2
1	3

6.3 Les sprites en mode 2

Le mode 2 comme le mode 1 permet de gérer 32 sprites (0 à 31).

Le VDP peut afficher 32 sprites sur ses 32 plans les plus prioritaires, à raison d'un sprite par plan.

Le nombre de sprites sur une même horizontale est limité à 8. La détection de présence d'un 9ème sprite est réalisée par le bit 6 du registre de statut S0. Les bits B0 à B4 donnent alors le numéro de ce 9ème sprite.

Lors d'une détection de collision, le bit 5 du registre de statut S0 est mis à 1.

La taille des sprites est gérée par le bit 1 (SI) sur registre R1.

SI = 0 : 8x8 points

SI = 1 : 16x16 points

Le bit 0 de R1 (MAG) permet de définir un facteur multiplicatif pour la taille des sprites.

MAG = 0 : Taille normale

MAG = 1 : Taille double

Chaque ligne horizontale d'un sprite peut contenir 2 couleurs.

La priorité des sprites peut être inhibée par l'intermédiaire du bit CC de la TAS (voir ci-après). Dans ce dernier cas, si deux sprites se recouvrent, un OU logique est réalisé entre les 2 couleurs des deux sprites. Autrement dit, 4 couleurs peuvent se retrouver à cet instant sur la même horizontale.

Les sprites sont définis par 3 tables : la TAS, la TGS et la TCS.

La TAS

La TAS spécifie les coordonnées et le numéro du sprite. La TAS comporte 4 octets par sprite.

Octet 1 : position verticale du point supérieur gauche. Si la valeur de cet octet est égale à 216, tous les sprites de priorité inférieure (numéro supérieur) ne seront pas affichés.

Octet 2 : position horizontale du coin supérieur gauche.

Octet 3 : pointeur relatif à la TGS. Il contient le numéro de matrice dans la TGS. En mode 16x16 il y a 4 matrices par sprite. Vous pouvez alors spécifier le numéro de la matrice de votre choix.

Octet 4 : réservé

Les 4 premiers octets de la TAS sont relatifs au sprite du plan 0, les 4 suivants au sprite du plan 1 et ainsi de suite jusqu'au plan 31. Il y a donc 128 octets dans la TAS.

LA TGS

La TGS est une table de 2048 octets organisée en 256 blocs de 8 octets. Elle définit le patron du sprite. La position de départ dans la TGS dans la VRAM est déterminée par le registre R6.

Le troisième octet de chaque entrée dans la TAS spécifie le bloc correspondant dans la TGS.

La TGS est organisée comme suit : les blocs de 8x8 octets sont groupés 4 par 4. Si le sprite est au format 8x8 ou 16x16 avec le bit d'agrandissement = 1, un seul bloc de 8 octets est occupé sur les 4, les 3 autres n'étant pas considérés par le VDP. Si le sprite est au format 16x16 avec le bit d'agrandissement à 0 ou au format 32x32, les 4 blocs sont visualisés dans l'ordre suivant :

0	2
1	3

Les bits à 0 dans la TGS dont toujours transparents, les bits à 1 sont en rapport avec la couleur spécifiée dans la TCS.

La TCS

La TCS est la table des couleurs des sprites. Elle est uniquement utilisée en mode 2. La couleur d'un sprite en mode 2 peut être différente sur chaque horizontale.

La TCS contient 16 octets par sprite. En mode 8x8 seuls les 8 premiers octets sont utilisés.

La TCS contient donc 512 (16*32) octets. La position de la TCS n'est pas fixée directement par un registre, elle est relative à la TAS. Le début de la TCS est toujours situé 512 octets avant le début de la TAS.

Chaque octet de la TAS a la même structure :

Bit	7	6	5	4	3	2	1	0
	EC	CC	IC	0	code de couleur			

Le code de couleur spécifie la couleur des bits à 1 dans la ligne horizontale correspondante dans la TGS.

EC Sémaphore de décalage (1 = +32 points)

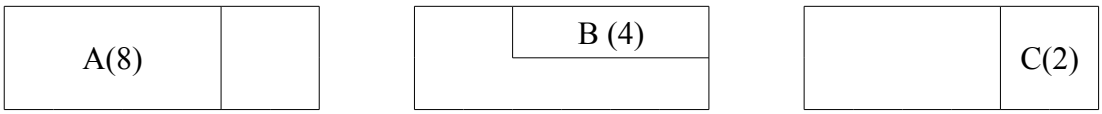
CC Sémaphore de priorité (1 = pas de priorité)

IC Sémaphore de détection de collision (1 = pas de détection)

Remarque : Si 2 sprites se rencontrent et qu'un seul à son bit IC à 1, la collision n'est pas détectée.

Exemple : affichage de sprites se recouvrant avec sémaphore de priorité positionné (priorité inhibée CC = 1).

Considérons 3 sprites A, B et C de couleurs respectives 8 (1000), 4 (0100) et 2 (0010) occupant les positions suivantes dans la même partie d'écran :



Résultat :

8	12	6
8	8	2

Si le sémaphore CC et le sémaphore IC sont à 0, la collision est détectée et les coordonnés de la collision se trouvent dans les registres d'état S3/4 (X) et S5/6 (Y). C'est la lecture du registre S5 qui remet les registres S3 et S6 à 0. Le registre S5 doit être lu en dernier.

Les coordonnées X et Y fournies doivent être augmentées d'un OFFSET (12 pour X et 8 pour Y)

afin de déterminer les coordonnées correctes de la collision.

Remarque sur les couleurs :

- Le bit TP (bit 5 de R8) peut affecter le traitement des bits à 0 dans la TGS.

  Si TP vaut 0, la couleur 0 est transparente, elle ne produit pas de collision.

  Si Tp vaut 1, la couleur 0 est mise dans la couleur de la palette correspondante. Dans ce cas la collision est détectée.

- En mode graphique 7, les sprites n'utilisent pas la palette de couleur, les couleurs sont fixées de façon interne (il y a donc 16 couleurs possibles). La table suivante fournit les valeurs des signaux RGB (rouge vert et bleu) pour les 16 possibilités.

Couleur	Rouge	Vert	Bleu
0	0	0	0
1	0	0	2
2	3	0	0
3	3	0	2
4	0	3	0
5	0	3	2
6	3	3	0
7	3	3	2
8	7	4	2
9	0	0	7
10	7	0	0
11	7	0	7
12	0	7	0
13	0	7	7
14	7	7	0
15	7	7	7

6.4 Déplacement des sprites

Nous avons vu que pour déplacer un sprite, il suffit de modifier les deux octets relatifs aux coordonnées de ce sprite dans la TAS.

Si une coordonnée a une valeur telle qu'une partie de ce sprite n'est pas dans la partie affichable de l'écran, la partie se trouvant dans l'écran est affichée normalement; celle qui est en dehors est cachée par le bord.

Ce système permet une apparition progressive des objets en mouvement. Pour faire sortir un sprite à droite ou en bas de l'écran, celui-ci doit avoir une coordonnée supérieure à 256 ou à 191 moins la taille du sprite (8 ou 16).

Pour faire apparaître ou sortir un sprite en haut à gauche de l'écran, le problème est plus ardu.

Pour la coordonnée verticale, pas de problème. Les coordonnées affichables étant comprises entre 0 et 191, il suffit de considérer l'octet de coordonnée verticale comme un octet exprimé en binaire signé si la valeur est supérieure à 191.

RAPPEL : Pour déterminer une valeur en binaire signé, il suffit d'ajouter 256 à la valeur désirée. Ainsi, -1 s'écrit -1 + 256 soit 255.

Exemple : une valeur de -3 (253) pour la coordonnée verticale permettra d'afficher un sprite en partie caché par le bord supérieur de l'écran.

La coordonnée verticale peut prendre toutes les valeurs comprises entre -31 (225) et +191. En effet,

il est inutile de dépasse -31 car à ce moment, un sprite de 32x32 points a complètement disparu de l'écran.

Pour la coordonnée horizontale, le problème est plus complexe. Les coordonnées affichables étant comprises entre 0 et 255, l'octet suffit tout juste pour définir la dite coordonnée.

L'astuce utilisée pour la coordonnée verticale n'est pas applicable. Pour réaliser ce tour de force, les concepteurs du VDP ont imaginé d'utiliser le bit le plus significatif du 4ème octet relatif au sprite concerné dans la TAS (l'octet qui définit la couleur).

Si ce bit est à 0, il n'influence pas l'affichage.

Si ce bit est à 1, la coordonnées horizontale (contenu du 2ème octet relatif au sprite concerné dans la TAS) est diminué de 32. La coordonnée horizontale du sprite peut donc être comprise entre -32 et +255.

7 Compléments

7.1 Le crayon optique

Pour utiliser un crayon optique, il faut que votre système soit initialement équipé du connecteur idoine.

1 - Positionner le bit 7 de R8 (MS) à 0 et le bit 6 de R8 à 1 (LP).

2 - Si vous désirez activer une interruption lorsque le crayon détecte une source lumineuse, positionnez le bit 5 de R0 à 1. L'interruption est remise à 0 à la lecture du registre d'état S1.

3 - S1 vous fournit 2 bits pour détecter la présence d'une source lumineuse ou l'appui sur le bouton du crayon : FL et LPS (bits 7 et 6).

4 - S3 et S4 vous fournissent la coordonnée X du point lumineux sur lequel se trouve le crayon.

5 - S5 et S6 vous fournissent la coordonnée Y du point lumineux sur lequel se trouve le crayon. Le bit 1 (Y9) de S6 indique le numéro du champ (0 = 1° champ, 1 = 2° champ).

6 - Les données de S3 à S6 restent stables tant que le registre S5 n'a pas été lu. Il faut donc terminer la lecture de la position par le registre S5.

7.2 La souris

La souris utilise le bus couleur. Lors de l'utilisation de la souris, le bus couleur ne peut pas être employé pour autre chose.

1 - Positionner le bit 7 de R8 (MS) à 1 et le bit 6 de R8 à 0 (LP).

2 - S1 vous fournit 2 bits pour détecter la pression sur les boutons de la souris : FL et LPS (bits 7 et 6).

3 - S3 vous fournit la coordonnée X de la position de la souris exprimée en complément à 2.

4 - S5 vous fournit la coordonnée Y de la position de la souris exprimée en complément à 2.

5 - Lorsque l'on écrit dans R15 (le registre qui contient le numéro de registre d'état) 3 ou 5, le comptage de la position de la souris n'est pas effectué.

6 - Après la lecture de S3 ou S5, la valeur de R15 doit être changée.

7.3 Synchronisation, superimposition...

La synchronisation permet le mixage de deux sources vidéo différentes. Elle est nécessaire pour permettre la convivialité des deux images.

La superimposition permet d'afficher une des deux images au dessus de l'autre. Elle permet de réaliser des opérations comme le titrage des films ou des effets spéciaux d'animation.

La digitalisation permet de transformer un signal vidéo extérieur en image informatique codée et donc traitable au moyen d'algorithmes divers. Cette dernière opération permet de traiter les images et de réaliser des vraies « œuvres d'art ». La digitalisation est possible en mode graphique 4 à 7.

Le système MSX2 Philips 8280 permet de réaliser toutes ces opérations avec une grande facilité. Les autres MSX2 pourraient réaliser le même miracle, mais il faut pour cela ajouter du matériel autour du VDP. La description de ce matériel sort malheureusement du cadre de cet ouvrage.

Les bits S0 et S1 de R9 permettent de choisir le mode de synchronisation.

S1	S0	Mode
0	0	Affiche l'écran MSX seul
0	1	Superimposition d'image ou digitalisation
1	0	Affichage de la source externe seule
1	1	Non utilisé

Pour permettre la digitalisation, il faut mettre le bit 6 du registre R0 (DG) à 1. A ce moment, le VDP peut lire des données en provenance du bus couleur.

Le bus couleur peut être masqué en utilisant le registre R7.

Si un bit de R7 est mis à 0, la valeur du bit correspondant sur le bus couleur est mise à 0.

En mode graphique 4 et 5, seuls les bits M0 à M3 (bit 0 à 3) de R7 sont utilisés. Ils sont en correspondance avec les bits C0 à C3 du bus couleur.

En mode graphique 6, tous les bits sont utilisés. M0 à M3 sont en correspondance avec C4 à C7 du bus couleur. M4 à M7 sont en correspondance avec C0 à C3 du bus couleur.

En mode graphique 7, tous les bits sont utilisés. M0 à M7 sont en correspondance avec C0 à C7 du bus couleur.

Le bus couleur est contrôlé par les bits MS (R2), DG (R0) et CB (R0).

MS	DG	CB	Fonction
0000	0000	0000	Fonctionnement normal
0	0	1	Affichage d'un signal vidéo extérieur
0	1	X	Digitalisation
1	X	X	Gestion de la souris par le bus

7.4 Affichage alternatif de deux pages écran

En mode graphique 4 à 7, deux pages écrans peuvent être affichées alternativement de façon automatique.

En mode graphique 4 et 5, on peut basculer de la page 0 à la page 1 et de la page 2 à la page 3.

En mode graphique 6 et 7, on peut basculer de la page 0 à la page 1.

Le registre R13 permet de sélectionner une période d'affichage variant entre 166 ms et 2053 ms par page.

Le bit E0 de R9 (bit 2) permet d'afficher alternativement deux pages à une vitesse de 60 Hz. La seconde page doit se trouver dans R2.

7.5 Utilisation des effets spéciaux en Basic

Il est possible de réaliser les effets spéciaux (digitalisation et incrustation) en Basic en utilisant une série d'instructions (OUT) qui positionnent les registres du VDP dans l'état ad hoc. Voici par exemple une méthode pour afficher l'image en provenance de la source extérieure.

```
10 REM PASSAGE IMAGE EXTERNE
20 OUT 247, 191
30 OUT 246, 223
40 COLOR 0, 0, 0
50 FOR I = 1 TO 3000 : NEXT I
60 REM RETOUR IMAGE NORMALE
70 OUT 247, 255
80 OUT 246, 255
90 COLOR 15, 4, 4
```

L'instruction SET VIDEO réalise cependant les mêmes opérations de façon beaucoup plus simple. La syntaxe utilise deux paramètres qui peuvent prendre respectivement les valeurs 0, 1, 2 et 0, 1.

Le bit 5 du port 0F6H permet le contrôle du mode mixage (arrêt/marche).

Si l'état des autres bits n'a pas d'importance, on peut utiliser :

```
ARRET : OUT &HF6, 255
MARCHE ; OUT &HF6, 223
```

Une inversion de l'état (marche/arrêt) peut toujours être obtenue sans modifier les autres bits par l'instruction

```
OUT &HF6, INP(&HF6) XOR 32
```

Le bit 6 du port F7 permet de sélectionner la source externe (vois structure de ce port dans le chapitre 1).

```
ARRET : OUT &HF7, 255
MARCHE : OUT &HF7, 191
```

Une inversion de l'état (marche/arrêt) peut toujours être obtenue sans modifier les autres bits par l'instruction

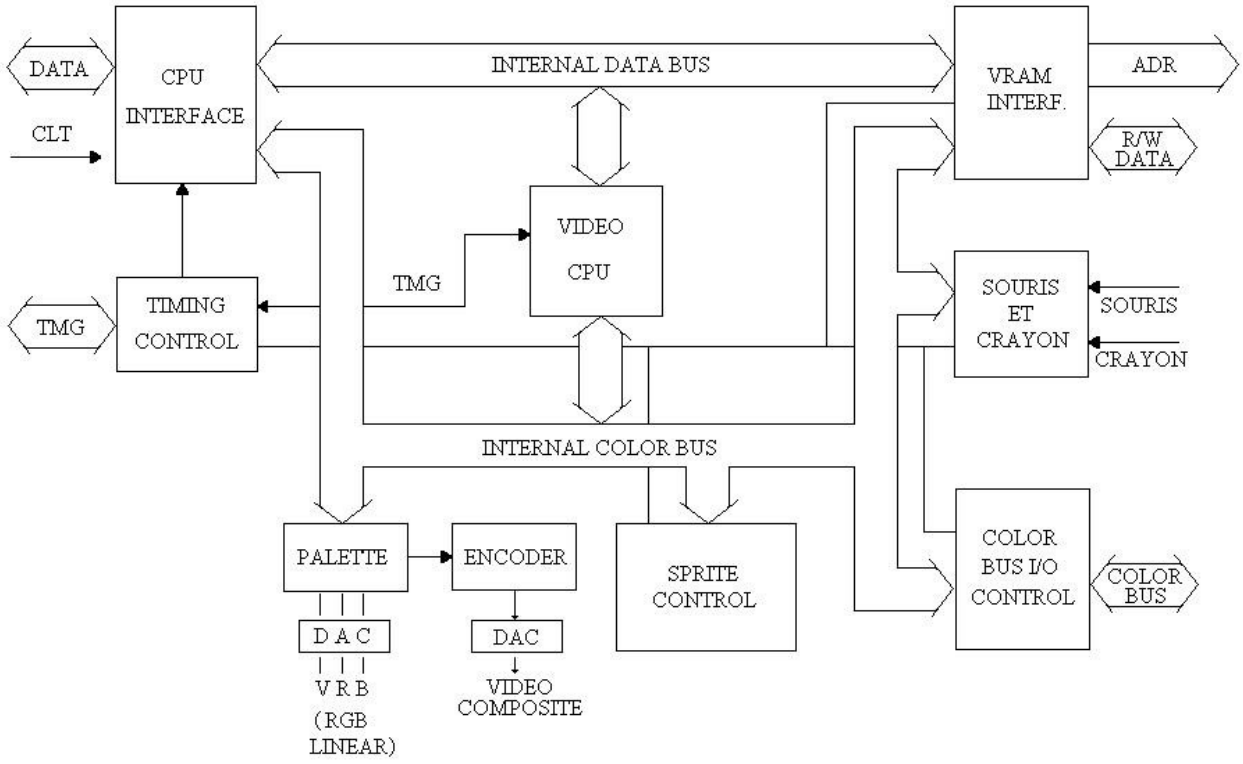
```
OUT &HF7, INP(&HF7) XOR 64
```

Pour rappel, voici une table des différentes possibilités de l'instruction SET VIDEO et du contrôle de mixage.

PARAMETRES	MIXAGE	SYNCHRO	FONCTION
0, 0	arrêt	interne	Graphisme seul luminosité 100 %
0, 0	marche	interne	Graphisme seul luminosité 50 %
1, 0	arrêt	externe	Graphisme seul luminosité 100 %
1, 0	marche	externe	Graphisme 50 % mixé à l'image externe 50 %
2, 0	arrêt	externe	Graphisme 100 % superposé à l'image externe 100 %
2, 0	marche	externe	Graphisme 50 % superposé à l'image externe 100 %
2, 1	arrêt	externe	Graphisme 100 % superposé à l'image externe 50 %
2, 1	arrêt	externe	Graphisme 50 % superposé à l'image externe 50 %

Pour la digitalisation, il faut utiliser le mode 1, 0 et l'instruction COPY SCREEN

```
10 SCREEN 8                    : HAUTE RÉSOLUTION
20 COLOR , , 255               : COULEUR TRANSPARENTE
30 SET VIDEO 1                       : MODE SYNCHRO EXTERIEURE
40 COPY SCREEN                 : DIGITALISATION
50 SET VIDEO 0, , 0            : REM RETOUR AU MODE NORMAL
```



CHAPITRE 4

AUTRES CIRCUITS

En plus du VDP et du PSG décrits dans les deux chapitres précédents, il reste deux circuits moins importants mais cependant intéressants à étudier, le PPI et l'horloge interne.

1 Le PPI

1.1 Généralités

Le PPI (Peripheral Port Interface) est un circuit fabriqué par Intel sous la dénomination 8255A. C'est un circuit d'interfaçage prévu pour les processeurs de la famille 8080.

Il possède 24 bits d'entrée/sortie qui peuvent être programmés en deux groupes de 12 bits et utilisés dans trois modes principaux.

Dans le premier mode (mode 0), chaque groupe de 12 bits peut ête programmé par tranche de 4 bits en entrée comme en sortie.

Dans le second mode (mode 1), chaque groupe de 12 bits peut être programmé de la façon suivante : 8 bits sont utilisés en entrée/sortie, les 4 autres sont utilisés pour le handshaking (contrôle de la transmission).

Le troisième mode (mode 2) est un mode où 8 bits sont utilisés comme port bidirectionnel et 5 bits pour le handshaking.

Le PPI possède aussi la possibilité de positionner des bits à l'état 1 ou 0 directement.

Pour plus de facilité, le PPI est divisé en 3 ports de 8 bits distincts appelés port A, port B et port C.

Le port C se divise en 2 groupes de 4 bits pour former les groupes de 12 bits avec A et B.

1.2 Découpage et utilisation des ports A, B, C

LE PORT A

Le port A est utilisé pour la commutation des slots mémoire.

La mémoire est divisée en 4 banks de 16 Ko :

BANK 0	adresse	0000 à 3FFF
BANK 1	adresse	4000 à 7FFF
BANK 2	adresse	8000 à BFFF
BANK 3	adresse	C000 à FFFF

Sur chaque bank, on peut sélectionner un slot parmi 4.

Les 8 bits du port A sont divisés de la façon suivante :

Bits 0 et 1 sélection du slot pour le bank 0

Bits 2 et 3 sélection du slot pour le bank 1

Bits 4 et 5 sélection du slot pour le bank 2

Bits 6 et 7 sélection du slot pour le bank 3

Chaque série permet 4 combinaisons :

00 01 10 11

En Basic, dans une configuration standard, tous les bits sont à 0 (les slots 0 sont sélectionnés).

LE PORT B

Le port B est très important car c'est lui qui permet la lecture du clavier. Les 8 bits sont utilisés uniquement en lecture pour déterminer si une touche est pressée ou non.

Nous nous attarderons par la suite sur le port B lors de l'analyse du fonctionnement du clavier.

LE PORT C

Le port C est divisé en 2 blocs de 4 bits numérotés B0-B3 et B4-B7.

Les bits B0-B3 sont utilisés uniquement en lecture et permettent de fournir l'adresse de la ligne des touches du clavier à lire.

Nous nous attarderons aussi par la suite sur le bloc B0-B3 du port C lors de l'analyse du fonctionnement du clavier.

Les bits B4-B7 sont utilisés pour la cassette et pour le signal SOUND.

Le bit B4 commande le démarrage ou l'arrêt de la cassette (signal CASON).

Le bit B5 commande l'écriture sur la cassette (signal CASWR).

Le bit B6 commande l'allumage de la lampe CAPS.

Le bit B7 commande le signal SOUND.

1.3 Programmation du PPI

Introduction

Le PPI est interfacé aux adresses suivantes :

A8H 168 : lecture et écriture du port A

A9H 169 : lecture du port B

AAH 170 : lecture et écriture du port C

ABH 171 : écriture du registre de contrôle

Remarque : de la configuration matérielle, nous pouvons déduire que le port B est utilisé en lecture uniquement, le registre de contrôle est utilisé en écriture uniquement et les ports A et C sont utilisés

dans les deux modes.

Des trois modes décrits brièvement dans les généralités, seul le mode 0 sera étudié car il suffit à toutes les manipulations envisagées.

Le PPI est programmable à travers un registre de contrôle dans lequel on ne peut qu'écrire. Aucune lecture de ce registre n'est permise.

Les ports du PPI doivent être divisés en deux groupes :

Le groupe A composé du port A et des 4 bits de poids fort du port C (B4-B7).

Le groupe B composé du port B et des 4 bits de poids faible du port C (B0-B3).

Le groupe B est entièrement réservé au clavier.

Le groupe A s'occupe de la gestion des slots, de la cassette, du son et de la touche CAPS.

Programmation

Ecriture dans le registre de contrôle

On écrit dans le registre de contrôle par une simple instruction OUT sur le port ABH du processeur.

Le mot de contrôle est un mot de 8 bits dont voici la signification bit par bit :

- Bit 7 : toujours 1 si c'est un mot de contrôle.
- Bit 6 : détermination du mode de fonctionnement du groupe A. Pour sélectionner le mode 0, ce bit doit être à 0. S'il est à l'état 1, il sélectionne le mode 2.
- Bit 5 : détermination du mode de fonctionnement du groupe A. Pour sélectionner le mode 0, ce bit doit être à 0. S'il est à l'état 1, il sélectionne le mode 1.
- Bit 4 : détermination du sens de fonctionnement du port A. 0 signifie en SORTIE et 1 signifie en ENTREE. Sera toujours 1.
- Bit 3 : détermination du sens de fonctionnement de la partie haute du port C. 0 signifie en SORTIE et 1 signifie en ENTREE.
- Bit 2 : détermination du mode de fonctionnement du groupe B. 0 signifie mode 0 et 1 signifie mode 1. Sera toujours 0.
- Bit 1 : détermination du sens de fonctionnement du port B.0 signifie en SORTIE et 1 signifie en ENTREE. Sera toujours 1.
- Bit 0 : détermination du sens de fonctionnement de la partie basse du port C. 0 signifie en SORTIE et 1 signifie en ENTREE. Sera toujours 0.

Si le bit 7 est égal à 0, le registre n'est plus utilisé en tant que contrôleur des ports, mais il permet de positionner les bits du port C à 1 ou à 0.

- Bit 7 = 0 : fonctionnement en positionnement de bits.
- Bits 6, 5 et 4 : non utilisés
- Bits 3, 2 et 1 : donnent le numéro de bit à positionner. Exemple : pour positionner le bit 5, mettre 1 dans B3, 0 dans B2 et 1 dans B1 car 101 donne 5.
- Bit 0 : donne le sens du positionnement. 1 signifie positionnement du bit à 1 et 0 signifie positionnement du bit à 0.

La programmation se fait donc en envoyant le mot d'état convenable sur le registre de contrôle et en effectuant une lecture ou une écriture sur le port idoine.

Exemple :

pour démarrer le moteur de la cassette, il suffit de mettre le bit 4 du port C à 0 :

10 OUT &HAB, &B00001000

Nous avons utilisé le mode de positionnement des bits. Analysons l'octet 00001000 :

Le premier bit en partant de la gauche (B7), égal à 0, signifie que l'on est en mode de positionnement de bits. Les 3 bits suivants (B4 à B6) ne servent à rien. Les bits B3 à B1 valent 100, c'est-à-dire 4. Donc c'est le bit 4 du port C qui est sélectionné. Le bit 0 vaut 0, ce qui signifie que le bit B4 du port C sera mis à 0.

1.4 Gestion du clavier

Le clavier est comparable à une matrice de 9 lignes et de 8 colonnes. Elle permet donc de disposer de 72 touches selon le schéma suivant :

PORT B= PORT C	B0	B1	B2	B3	B4	B5	B6	B7
Ligne 0	0	1	2	3	4	5	6	7
Ligne 1	8	9	-	+	\	[	]	;
Ligne 2	"	£	<	>	?	ACC	A	B
Ligne 3	C	D	E	F	G	H	I	J
Ligne 4	K	L	M	N	O	P	Q	R
Ligne 5	S	T	U	V	W	X	Y	Z
Ligne 6	SHIFT	CTRL	GRAPH	CAP	CODE	F1	F2	F3
Ligne 7	F4	F5	ESC	TAB	STOP	BS	SEL	ENTER
Ligne 8	SPC	CLS	INS	DEL	fl.G	fl.H	fl.B	fl.D

Remarque : une seule des gravures des touches a été représentée.

Les abréviations suivantes ont été utilisées :

ACC = touche accent à côté de l'accolade droite

SEL = touche SELECT

SPC = barre d'espacement

fl.G = flèche gauche

fl.D = flèche droite

fl.H = flèche haut

fl.B = flèche bas

Pour lire cette matrice, on transmet sur la partie basse du port C (B0-B3) le numéro de la ligne à scruter. Ensuite, on lit le contenu du port B.

Un bit est à 0 si la touche correspondante est enfoncée, sinon il est à 1.

Les bits du port B étant numérotés de B0 à B7, si l'on appuie sur la touche A par exemple, le bit B6 passe à 0 et les autres bits restent à 1.

Le bit B6 correspondant à la touche A est lisible uniquement lorsque le numéro de la ligne (en l'occurrence 2) a été inscrit sur le port C.

Exemple : pour décoder l'appui sur la touche SELECT (non utilisé par le Basic), il faut :

- écrire le numéro de ligne dans le port C (ligne 7).
- lire le contenu du port B
- masquer les bits inutiles
- tester le bit concerné (B6)

C'est le but du petit programme suivant :

```
10 OUT &HAA, 7 : ' ÉCRITURE DU NUMÉRO DE LIGNE DANS C
20 X = INP(&HA9) : 'LECTURE DU PORT B
30 X = X AND 64 : 'MASQUAGE DES BITS DIFFÉRENTS DE B6
40 IF X = 0 THEN PRINT « TOUCHE SELECT ENFONCÉE »
50 GOTO 10
```

2 L'horloge interne RP5C01

2.1 Généralités

Le circuit d'horloge qui équipe les MSX2 est référencé sous le numéro RP5C01. Il se compose d'un circuit d'horloge très semblable à celui d'une montre. Ce circuit reste alimenté par une batterie interne même si la tension générale du système est coupée. Il permet donc de disposer d'un repère temporel exact à l'allumage du système.

Ce circuit possède une mémoire connexe sauvegardée sur batterie. Le système MSX2 utilise cette mémoire pour sauvegarder différents paramètres initialisés lors de l'allumage du système. Elle permet principalement de sauvegarder de façon permanente le mot de passe et les paramètres de l'écran.

Le circuit est contrôlé par 2 ports (B4H et B5H) qui permettent de sélectionner 4 modes de fonctionnement possibles.

Les modes 0 et 1 permettent de positionner l'heure, la date et une alarme éventuelle (l'alarme n'est pas gérée par le système MSX).

Les modes 2 et 3 sont utilisés pour mémoriser l'état interne du système MSX dans la RAM sauvegardée sur batterie.

La mémoire peut être divisée en 4 blocs de 16 octets. Chaque bloc est actif dans un mode précis.

Ce circuit est programmable par l'intermédiaire des vecteurs 01F5H et 01F9H de la ROM d'extension MSX2.

La routine 1F5H permet de lire la valeur d'une mémoire dont l'adresse est préalablement disposée dans le registre C. La valeur lue se retrouve dans l'accumulateur A.

La routine 1F9H permet d'écrire dans une mémoire du circuit, une valeur contenue dans le registre A à l'adresse contenue dans le registre C.

La structure du registre C est la suivante :

B7	B6	B5	B4	B3	B2	B1	B0
X	X	M1	M0	A3	A2	A1	A0

M0 et M1 déterminent les 4 modes.

A0 à à A3 déterminent l'adresse à utiliser (0 à 15)

Les valeurs lues ou écrites sont des valeurs sur 4 bits (les moins significatifs).

2.2 Contenu de la mémoire du circuit horloge

En mode 2

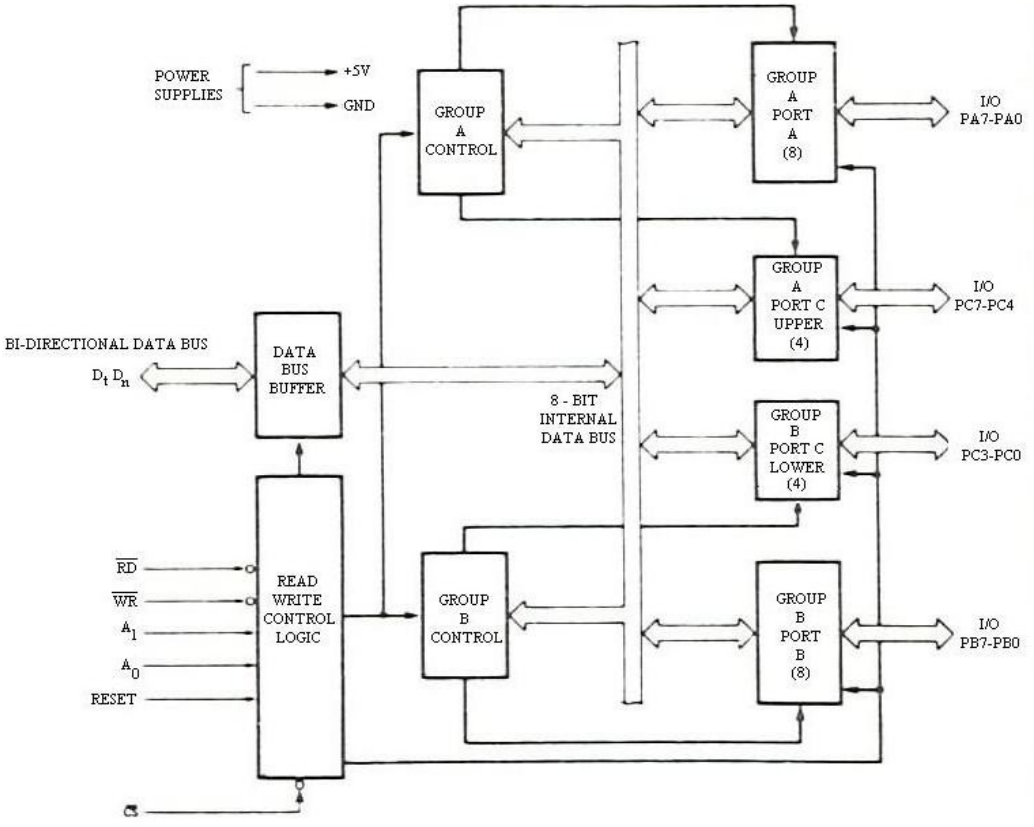
ADR	Instruction	Contenu des 4 bits
0		1/2 octet ID (1010 indique que la RAM horloge est initialisée)
1	SET ADJUST	Ajuste X (-8 à 7)
2	SET ADJUST	Ajuste Y (-8 à 7)
3	SET SCREEN	Mode d'écran 0 ou 1, mode entrelacé <==2 bits à gauche
4	SET SCREEN	Octet bas du WIDTH
5	SET SCREEN	Octet haut du WIDTH
6	SET SCREEN	Couleur d'avant-plan (0-15)
7	SET SCREEN	Couleur d'arrière-plan (0-15)
8	SET SCREEN	Couleur du bord (0-15)
9	SET SCREEN	Bruit accompagnant l'enfoncement d'une touche Touches de fonction activées ou désactivées Mode de l'imprimante Vitesse du lecteur de cassettes
10	SET BEEP	Sélection du son BEEP Volume de celui-ci (0-3, 0-3)
11	SET TITLE	Couleur du titre (0-3) <==2 bits à gauche
12		Code du pays (0-15) • Table des codes de pays 0 - Japon 1 - Etats-Unis 2 - Grande-Bretagne 3 - France 4 - Allemagne 5 - Italie 6 - Espagne 7 - Arabie 8 - Corée 9 - 10 - 11 - 12 - 13 - 14 - 15 -

En mode 3

ADR	Contenu des 4 bits		
0	1/2 octet ID	0 : Titre,	1 : Mot de passe 2 : Indicatif 3-15 : Non définis
	1	Faible1 -------+	
	2	Fort1 |	
	3	Faible2 |	
	4	Fort2 |	
	5	Faible3 |	Chaîne de datas de 6 octets pour usage
	6	Fort3 |	indiqué ci-dessus
	7	Faible4 |	
	8	Fort4 |	
	9	Faible5 |	
	10	Fort5 |	
	11	Faible6 |	
	12	Fort6 ----------+	

Usage de la RAM pour le mot de passe

adresse 0	- usage id = 1
adresses 1, 2, 3	- usage id = 1, 2, 3
adresses 4, 5, 6, 7	- mot de passe encodé
adresse 8	- sémaphore vérifiant l'existence d'une cartouche-clé : = 0 - pas de cartouche-clé = 1 - présence d'une cartouche, aucun mot de passe positionné = 2 - présence d'une cartouche, mot de passe positionné
adresses 9, 10, 11, 12	- valeur des cartouches-clé



CHAPITRE 5

POINTS D'ENTRÉE DU BIOS ET TABLE DES VARIABLES

Le but de ce chapitre est de vous fournir la liste exhaustive des routines du Bios et la table des variables internes. C'est en adressant ces routines par l'intermédiaire de leurs vecteurs d'appel situés en début de mémoire que vous assurerez une portabilité et une compatibilité parfaite de vos logiciels.

1 Points d'entrée de la ROM principale

ROM principale (Slot 0/ 0-0, pages 0 et 1)

Réservation des RST :

0 à 5	→	interpréteur Basic
6	→	appels inter-slots
7	→	interruptions matérielles

Signification des notes utilisées :

Adresse	→	Adresse d'exécution de la fonction
Nom	→	Nom de la fonction
Fonction	→	Utilisation de la fonction
C. E.	→	Conditions d'entrée
C. S.	→	Conditions de sortie
R.M.	→	Registre(s) modifié(s)
Rem	→	Remarques éventuelles

Certaines entrées sont envoyées à la Sub-ROM (ROM complémentaire).

*1	Sans changement par rapport au MSX1
*2	Envoi à la Sub-ROM si SCREEN 5 à 8
*3	Envoi à la Sub-ROM
*4	Pas d'envoi, mais la routine est transformée en mode SCREEN 4 à 8.

Remarque : Les conditions d'entrée et de sortie concernent les registres internes du Z80. Lorsqu'il n'y a pas de conditions indiquées et que la routine n'est pas triviale, c'est que la routine utilise des variables internes.

Adresse : 0000H
Nom : **CHKRAM** *1
Fonction : Teste le RAM et positionne le slot correct
C. E. : Sans objet
C. S. : Sans objet
R.M. : Tous
Rem : Quand cette fonction est terminée, il faut réaliser un saut à INIT pour une initialisation complète.

Adresse : 0008H
Nom : **SYNCHR** *1
Fonction : Teste si le caractère courant, pointé par HL est bien le caractère attendu. Si ce n'est pas le cas, un message « Syntax error » est généré; autrement, un saut à CHRGTR est exécuté.
C. E. : HL pointe sur le caractère à tester, le caractère de comparaison est placé à l'adresse qui suit le RST.
C. S. : HL pointe vers le caractère suivant, A contient le caractère testé. Le sémaphore de retenue (Carry Flag) est mis à 1, s'il s'agit d'un nombre. Le sémaphore de zéro est mis à 1, si c'est la fin d'une instruction.
R.M. : AF, HL

Adresse : 000CH
Nom : **RDSLT** *1
Fonction : Sélectionne le slot approprié en accord avec la valeur transmise par les registres et lit le contenu de la mémoire à partir du slot ainsi sélectionné.
C. E. : A = FxxxSSPP
| || ||
| || - - - - - slot primaire # (0-3)
| - - - - - - - slot secondaire # (0-3)
- - - - - - - - - - - 1 si le slot secondaire est spécifié
HL contient l'adresse de la mémoire cible
C. S. : A contient le contenu de la mémoire
R.M. : AF, BC, DE
Rem : Les interruptions sont mises automatiquement hors de service, mais ne sont jamais réactivées par cette routine.

Adresse : 0010H
Nom : **CHRGTR** *1
Fonction : Lit le caractère suivant (ou le mot-clé) à partir du texte Basic.
C. E. : HL
C. S. : HL pointe vers le caractère suivant
A contient le caractère lu
Le sémaphore de retenue est mis à 1 si c'est un nombre
Le sémaphore de 0 est mis à 1 si la fin de l'instruction est rencontrée
R.M. : AF, HL

Adresse : 0014H
Nom : **WRSLT** *1
Fonction : Sélectionne le slot approprié et écrit un octet à une adresse précise de la mémoire ainsi sélectionnée.
C. E. : A = FxxxSSPP
| || ||
| || - - - - - slot primaire # (0-3)
| - - - - - - - slot secondaire # (0-3)
- - - - - - - - - - - 1 si le slot secondaire est spécifié
HL contient l'adresse de la mémoire cible
E contient la donnée à écrire
C. S. : Sans objet
R.M. : AF, BC, D
Rem : Les interruptions sont mises automatiquement hors de service, mais ne sont jamais réactivées par cette routine.

Adresse : 0018H
Nom : **OUTDO** *2
Fonction : Sortie vers le périphérique en service
C. E. : A contient le caractère à écrire
PTRFIL et PRTFLG permettent de décider de l'orientation du caractère à écrire (écran ou imprimante)
C. S. : Sans objet
R.M. : Sans objet

Adresse : 001CH
Nom : **LSLT** *1
Fonction : Appel inter-slots à l'adresse spécifiée
C. E. : IYH = FxxxSSPP (partie haute du registre IY)
| || ||
| || - - - - - slot primaire # (0-3)
| - - - - - - - slot secondaire # (0-3)
- - - - - - - - - - - 1 si le slot secondaire est spécifié
IX contient l'adresse à appeler
C. S. : Inconnu
R.M. : Inconnu
Rem : Les interruptions sont mises automatiquement hors service, mais ne sont jamais réactivées par cette routine. Vous ne pouvez jamais passer des arguments par l'intermédiaire des registres secondaires du Z80 ou par IX et IY.

Adresse : 0020H
Nom : **DCOMPR** *1
Fonction : Compare HL avec DE
C. E. : HL, DE
C. S. : si HL = DE, le sémaphore de zéro est positionné à 1
si HL< DE, le sémaphore de retenue est mis à1
si HL>DE, le sémaphore de retenue est mis à 0
R.M. : AF

Adresse : 0024H
Nom : **ENASLT** *1
Fonction : Sélectionne le slot approprié en accord avec la valeur transmise pr les registres et active le slot en permanence.
C. E. : A = FxxxSSPP

```
|    || ||
|    || - - - - - slot primaire # (0-3)
|    - - - - - - - slot secondaire # (0-3)
- - - - - - - - - - 1 si le slot secondaire est spécifié
```

HL contient l'adresse de la mémoire cible
C. S. : Sans objet
R.M. : Tous
Rem : Les interruptions sont mises automatiquement hors service, mais ne sont jamais réactivées par cette routine.

Adresse : 0028H
Nom : **GETYPR** *1
Fonction : renvoie le type de FAC (accumulateur mathématique)
C. E. : FAC
C. S. : Type entier : sémaphore de signe = 1
Type chaîne de caractères : sémaphore de zéro = 1
Type simple précision : sémaphore de parité = 1
Type double précision : sémaphore de retenue = 1
R.M. : AF

Adresse : 0030H
Nom : **CALLF** *1
Fonction : Réalise l'appel distant (appel inter-slot)
C. E. : Sans objet
C. S. : Inconnu
R.M. : Inconnu
Rem : La séquence est appelée comme suit :

```
RST   6
DB    Slot de destination (1 octet)
DW    Adresse de destination (2 octets)
```

Adresse : 0038H
Nom : **KEYINT** *1
Fonction : Effectue les procédures d'interruptions nécessaires
C. E. : Sans objet
C. S. : Sans objet
R.M. : Sans objet
Rem : Appelle la ROM étendue pour la conversion ROMA-KANA

Adresse : 0041H
Nom : **DISSCR** *1
Fonction : Met l'affichage écran hors-service
C. E. : Sans objet
C. S. : Sans objet
R.M. : AF, BC

Adresse : 0044H
Nom : **ENASCR** *1
Fonction : Met l'affichage écran en service
C. E. : Sans objet
C. S. : Sans objet
R.M. : AF, BC

Adresse : 0047H
Nom : **WRTVDP** *2
Fonction : Ecrit dans un des registres du VDP
C. E. : C contient le numéro du registre, B contient la donnée à écrire. Le numéro de registre doit être compris entre 0 et 23 ou 32 et 46.
C. S. : Sans objet
R.M. : AF, BC
Rem : Appelle la ROM étendue si :
- le bit EV du registre 0 est modifié
- les registres 8 à 46 sont utilisés.

Adresse : 004AH
Nom : **RDVRM** *1
Fonction : Lit la VRAM adressée par HL. Les adresses sont limitées à 16Ko (A0 à A13). Pour utiliser 64 Ko, il faut utiliser NRDVRM.
C. E. : HL
C. S. : A contient la valeur lue
R.M. : AF

Adresse : 004DH
Nom : **WRTVRM** *1
Fonction : Ecrit dans la VRAM adressée par HL. Les adresses sont limitées à 16Ko (A0 à A13). Pour utiliser 64 Ko, il faut utiliser NWRVRM.
C. E. : HL, A contient l'octet à écrire
C. S. : Sans objet
R.M. : AF

Adresse : 0050H
Nom : **SETRD** *1
Fonction : Prépare le VDP pour la lecture
C. E. : HL contient l'adresse. Les adresses sont limitées à 16 Ko (A0 à A13). Pour utiliser 64 Ko, il faut utiliser NSETRD.
C. S. : Sans objet
R.M. : AF

Adresse : 0053H
Nom : **SETWRT** *1
Fonction : Prépare le VDP pour l'écriture
C. E. : HL contient l'adresse. Les adresses sont limitées à 16 Ko (A0 à A13). Pour utiliser 64 Ko, il faut utiliser NSTWRT.
C. S. : Sans objet
R.M. : AF

Adresse : 0056H
Nom : **FILVRM** *4
Fonction : Remplit la VRAM avec la donnée spécifiée
C. E. : HL contient l'adresse
BC contient la longueur (nombre d'octets)
A contient la donnée
Pour utiliser 64 Ko il faut appeler BIGFIL
C. S. : Sans objet
R.M. : AF, BC

Adresse : 0059H
Nom : **LDIRMV** *4
Fonction : Déplace un bloc de mémoire de la VRAM vers la mémoire centrale.
C. E. : HL contient l'adresse source (tous les bits sont valides)
DE contient l'adresse de destination
BC contient la longueur
C. S. : Sans objet
R.M. : Tous

Adresse : 005CH
Nom : **LDIRVM** *4
Fonction : Déplace un bloc de mémoire de la mémoire centrale vers la VRAM.
C. E. : HL contient l'adresse source
DE contient l'adresse de destination (tous les bits sont valides)
BC contient la longueur
C. S. : Sans objet
R.M. : Tous

Adresse : 005FH
Nom : **CHGMOD** *3
Fonction : Sélectionne le mode VDP en accord avec SCRMOD (mode écran).
La palette n'est pas initialisée.
Pour initialiser la palette, il faut appeler CHGMDP dans la ROM étendue.
C. E. : A contient le mode écran (0 à 8)
C. S. : Sans objet
R.M. : Tous

Adresse : 0062H
Nom : **CHGCLR** *1
Fonction : Change la couleur de l'écran.
C. E. : A contient le mode
FORCLR contient la couleur d'avant-plan
BAKCLR contient la couleur d'arrière-plan
BDRCLR contient la couleur du bord
C. S. : Sans objet
R.M. : Tous

Adresse : 0066H
Nom : **NMI** *1
Fonction : Effectue une procédure d'interruption non masquable
C. E. : Sans objet
C. S. : Sans objet
R.M. : Sans objet

Adresse : 0069H
Nom : **CLRSPR** *3
Fonction : Initialise tous les sprites.
Les patrons sont mis à zéro.
Le nom de chaque sprite est égal au numéro du plan du sprite.
La couleur de chaque sprite est égale à la couleur d'avant-plan
La position verticale est positionnée à 209. Si le mode écran est compris entre 4 et 8, la position verticale est alors égale à 217.
C. E. : SCRMOD
C. S. : Sans objet
R.M. : Tous

Adresse : 006CH
Nom : **INITXT** *3
Fonction : Initialise l'écran pour le mode texte (40x24).
Positionne le VDP.
Cette routine n'initialise pas la palette. Pour ce faire, il faut appeler INIPLT dans la ROM étendue après cet appel-ci.
C. E. : TXTNAM, TXTCGP
C. S. : Sans objet
R.M. : Tous

Adresse : 006FH
Nom : **INIT32** *3
Fonction : Initialise l'écran pour le mode texte (32x24).
Positionne le VDP.
Cette routine n'initialise pas la palette.
C. E. : T32NAM, T32CGP, T32COL, T32ATR, T32PAT
C. S. : Sans objet
R.M. : Tous

Adresse : 0072H
Nom : **INIGRP** *3
Fonction : Initialise l'écran pour le mode haute-résolution.
Positionne le VDP.
Cette routine n'initialise pas la palette.
C. E. : GRPNAM, GRPCGP, GRPCOL, GRPATR, GRPPAT
C. S. : Sans objet
R.M. : Tous

Adresse : 0075H
Nom : **INIMLT** *3
Fonction : Initialise l'écran pour le mode multicolore.
Positionne le VDP.
Cette routine n'initialise pas la palette.
C. E. : MLTNAM, MLTCGP, MLTCOL, MLTATR, MLTPAT
C. S. : Sans objet
R.M. : Tous

Adresse : 0078H
Nom : **SETTXT** *3
Fonction : Met le VDP à 1 pour le mode texte (40x24).
C. E. : TNTNAM, TXTCGP
C. S. : Sans objet
R.M. : Tous

Adresse : 007BH
Nom : **SETT32** *3
Fonction : Positionne le VDP pour le mode texte (32x24).
C. E. : T32NAM, T32CGP, T32COL, T32ATR, T32PAT
C. S. : Sans objet
R.M. : Tous

Adresse : 007EH
Nom : **SETGRP** *3
Fonction : Positionne le VDP pour le mode haute résolution.
C. E. : GRPNAM, GRPCGP, GRPCOL, GRPATR, GRPPAT
C. S. : Sans objet
R.M. : Tous

Adresse : 0081H
Nom : **SETMLT** *3
Fonction : Positionne le VDP pour le mode multicolore.
C. E. : MLTNAM, MLTCGP, MLTCOL, MLTATR, MLTPAT
C. S. : Sans objet
R.M. : Tous

Adresse : 0084H
Nom : **CALPAT** *1
Fonction : Retourne l'adresse d'un sprite dans la table des patrons.
C. E. : A contient l'identificateur du sprite.
C. S. : HL contient l'adresse du sprite dans la table.
R.M. : AF, DE, HL

Adresse : 0087H
Nom : **CALATR** *1
Fonction : Renvoie l'adresse d'un sprite dans la table d'attribut de sprite.
C. E. : A contient l'identificateur du sprite.
C. S. : HL contient l'adresse du sprite dans la table.
R.M. : AF, DE, HL

Adresse : 008AH
Nom : **GSPSIZ** *1
Fonction : Fournit la taille courante du sprite.
C. E. : Sans objet
C. S. : A contient la taille du sprite. Si taille = 16x16, le sémaphore de retenue est mis à 1.
R.M. : AF

Adresse : 008DH
Nom : **GRPPRT** *2
Fonction : Imprime un caractère sur l'écran graphique.
C. E. : A contient le code du caractère à imprimer.
Si le mode écran est compris entre 5 et 8 : (LOGOPR) contient le code d'opération logique.
C. S. : Sans objet
R.M. : Sans objet
Rem : Pour un mode écran compris entre 5 et 8, il y a un appel à la ROM étendue.

Adresse : 0090H
Nom : **GICINI** *1
Fonction : Initialise le PSG, et les données résidentes pour l'instruction PLAY.
C. E. : Sans objet
C. S. : Sans objet
R.M. : Tous

Adresse : 0093H
Nom : **WRTPSG** *1
Fonction : Ecrit une donnée dans un registre du PSG
C. E. : A contient le numéro du registre
E contient la donnée
C. S. : Sans objet
R.M. : Sans objet

Adresse : 0096H
Nom : **RDPSG** *1
Fonction : Lit une donnée à partir d'un registre du PSG.
C. E. : A contient le numéro du registre
C. S. : A contient la donnée lue
R.M. : Sans objet

Adresse : 0099H
Nom : **STRTMS** *1
Fonction : Teste et fait démarrer la tâche d'arrière-plan de PLAY.
C. E. : Sans objet
C. S. : Sans objet
R.M. : Tous

Adresse : 009CH
Nom : **CHSNS** *1
Fonction : Teste l'état du buffer du clavier
C. E. : Sans objet
C. S. : Le sémaphore de zéro est mis à 0 s'il y a un caractère dans le buffer.
R.M. : AF

Adresse : 009FH
Nom : **CHGET** *1
Fonction : Attend jusqu'à ce qu'un caractère soit entré au clavier et retourne avec le code de ce caractère.
C. E. : Sans objet
C. S. : A contient le code du caractère
R.M. : AF

Adresse : 00A2H
Nom : **CHPUT** *1
Fonction : Sort un caractère vers la console.
C. E. : A contient le code du caractère à sortir
C. S. : Sans objet
R.M. : Sans objet

Adresse : 00A5H
Nom : **LPTOUT** *1
Fonction : Sort un caractère vers l'imprimante
C. E. : A contient le code du caractère à écrire
C. S. : Le sémaphore de retenue est mis à 1 si l'opération échoue.
R.M. : F

Adresse : 00A8H
Nom : **LPTSTT** *1
Fonction : Teste l'état de l'imprimante
C. E. : Sans objet
C. S. : A contient 255 et le sémaphore de zéro est mis à 0 si l'imprimante est prête.
A contient 0 et le sémaphore de zéro est mis à 1 si l'imprimante n'est pas prête.
R.M. : AF

Adresse : 00ABH
Nom : **CNVCHR** *1
Fonction : Teste l'octet de l'en-tête graphique et le code de conversion.
C. E. : Code du caractère dans A
C. S. : Si le code correspond à un octet d'en-tête graphique, le sémaphore de retenue est mis à 0.
Si le code correspond à un code graphique converti, le sémaphore de retenue et le sémaphore de zéro sont mis à 1.
Si le code correspond à un code graphique non converti, le sémaphore de retenue est mis à 1 et le sémaphore de zéro est mis à 0.
R.M. : AF

Adresse : 00AEH
Nom : **PINLIN** *1
Fonction : Accepte une ligne à partir du clavier jusqu'à ce qu'un RETURN (CR) ou un STOP soit frappé et charge cette ligne dans le buffer.
C. E. : Sans objet
C. S. : HL contient l'adresse supérieure -1 du buffer.
Le sémaphore de retenue est mis à 1 si un STOP est frappé.
R.M. : Tous

Adresse : 00B1H
Nom : **INLIN** *1
Fonction : Identique à PINLIN sauf si AUTFLG est mis à 1.
C. E. : Sans objet
C. S. : HL contient l'adresse supérieure -1 du buffer.
Si STOP : le sémaphore de retenue est mis à 1.
R.M. : Tous

Adresse : 00B4H
Nom : **QINLIN** *1
Fonction : Sort un '?' suivi d'un espace puis saute à INLIN.
C. E. : Sans objet
C. S. : HL contient l'adresse supérieure -1 du buffer.
Si STOP : le sémaphore de retenue est mis à 1.
R.M. : Tous

Adresse : 00B7H
Nom : **BREAKX** *1
Fonction : Teste l'état de CTRL-STOP.
C. E. : Sans objet
C. S. : Le sémaphore de retenue est mis à 1 si CTRL-STOP est frappé.
R.M. : AF
Rem : Cette routine est utilisée pour tester CTRL-STOP quand les interruptions sont hors-service.

Adresse : 00BAH
Nom : **ISCNTC** *1
Fonction : Teste l'état de SHIFT-STOP.
C. E. : Sans objet
C. S. : Sans objet
R.M. : Sans objet

Adresse : 00BDH
Nom : **CKCNTC** *1
Fonction : Comme ISCNTC mais utilisé en Basic.
C. E. : Sans objet
C. S. : Sans objet
R.M. : Sans objet

Adresse : 00C0H
Nom : **BEEP** *3
Fonction : Emet un 'beep'.
C. E. : Sans objet
C. S. : Sans objet
R.M. : Tous

Adresse : 00C3H
Nom : **CLS** *3
Fonction : Nettoie l'écran
C. E. : Le sémaphore de zéro doit être mis à 1.
C. S. : Sans objet
R.M. : AF, BC, DE

Adresse : 00C6H
Nom : **POSIT** *1
Fonction : Place le curseur à la position spécifiée.
C. E. : H contient le numéro de colonne.
L contient le numéro de ligne.
C. S. : Sans objet
R.M. : AF

Adresse : 00C9H
Nom : **FNKSB** *1
Fonction : Teste si l'affichage des touches de fonction est actif. Si oui, l'affichage est effectué, si non, l'effacement de la ligne est effectué.
C. E. : FNKFLG (0F3DEH)
C. S. : Sans objet
R.M. : Tous

Adresse : 00CCH
Nom : **ERAFNK** *1
Fonction : Efface la ligne d'affichage des touches de fonction.
C. E. : Sans objet
C. S. : Sans objet
R.M. : Tous

Adresse : 00CFH
Nom : **DSPFNK** *2
Fonction : Affiche la ligne d'affichage des touches de fonction.
C. E. : Sans objet
C. S. : Sans objet
R.M. : Tous

Adresse : 00D2H
Nom : **TOTEXT** *1
Fonction : Force l'écran en mode texte
C. E. : Sans objet
C. S. : Sans objet
R.M. : Tous
Rem : TOTEXT n'est pas changé, mais cette routine appelle CHGMDP.

Adresse : 00D5H
Nom : **GTSTCK** *1
Fonction : Renvoie l'état courant de la manette de jeu (joystick).
C. E. : A contient le numéro de la manette de jeu.
C. S. : A contient la direction de la manette.
R.M. : Tous

Adresse : 00D8H
Nom : **GTTRIG** *1
Fonction : Renvoie l'état courant du bouton de tir.
C. E. : A contient le numéro du bouton de tir.
C. S. : Renvoie 0 dans A si le bouton de tir n'est pas pressé, 255 dans le cas contraire.
R.M. : AF

Adresse : 00DBH
Nom : **GTPAD** *1
Fonction : Teste l'état courant du PAD (tablette graphique).
C. E. : A contient le numéro du PAD
C. S. : A contient la valeur lue
R.M. : Tous

Adresse : 00DEH
Nom : **GTPDL** *2
Fonction : Renvoie la valeur de la manette analogique (paddle).
C. E. : A contient le numéro de la manette analogique.
C. S. : A contient la valeur lue
R.M. : Tous

Adresse : 00E1H
Nom : **TAPION** *1
Fonction : Met l'enregistreur en marche et lit l'en-tête à partir de la bande.
C. E. : Sans objet
C. S. : Le sémaphore de report (CARRY) est mis à 1 si l'opération échoue
R.M. : Tous

Adresse : 00E4H
Nom : **TAPIN** *1
Fonction : Lecture de bande cassette.
C. E. : Sans objet
C. S. : A contient la donnée lue.
Le sémaphore de report est mis à 1 si l'opération échoue.
R.M. : Tous

Adresse : 00E7H
Nom : **TAPIOF** *1
Fonction : Arrête la lecture de la bande cassette
C. E. : Sans objet
C. S. : Sans objet
R.M. : Sans objet

Adresse : 00EAH
Nom : **TAPOON** *1
Fonction : Met l'enregistreur en marche et écrit le bloc en-tête sur la bande cassette.
C. E. : A contient une valeur différente de 0 si un grand en-tête est désiré, 0 si un petit en-tête est désiré.
C. S. : Le sémaphore de report est mis à 1 si l'opération échoue.
R.M. : Tous

Adresse : 00EDH
Nom : **TAPOUT** *1
Fonction : Ecrit les données sur la bande cassette.
C. E. : A contient la donnée à écrire.
C. S. : Le sémaphore de report est mis à 1 si l'opération échoue.
R.M. : Tous

Adresse : 00F0H
Nom : **TAPOOF** *1
Fonction : Arrête l'écriture sur la bande cassette.
C. E. : Sans objet
C. S. : Le sémaphore de report est mis à 1 si l'opération échoue.
R.M. : Sans objet

Adresse : 00F3H
Nom : **STMOTR** *1
Fonction : Arrête ou démarre le moteur du lecteur de cassettes.
C. E. : 0 dans A pour arrêter.
1 dans A pour démarrer.
255 dans A pour inverser le mode.
C. S. : Sans objet
R.M. : AF

Adresse : 00F6H
Nom : **LFTQ** *1
Fonction : Renvoie le nombre d'octets en file d'attente.
C. E. : Sans objet
C. S. : A contient le nombre d'octets
R.M. :

Adresse : 00F9H
Nom : **PUTQ** *1
Fonction : Ajoute un octet à la file d'attente.
C. E. : E contient l'octet à ajouter
C. S. : Si la file est pleine, le sémaphore de zéro est mis à 1.
R.M. :

Adresse : 00FCH
Nom : **RIGHTC** *2
Fonction : Déplace l'accumulateur graphique d'un pixel à droite.
C. E. :
C. S. : Si le bord est atteint, le CARRY est positionné à 1.
R.M. :

Adresse : 00FFH
Nom : **LEFTC** *2
Fonction : Déplace l'accumulateur graphique d'un pixel à gauche.
C. E. :
C. S. : Si le bord est atteint, le sémaphore de retenue est positionné à 1.
R.M. :

Adresse : 0102H
Nom : **UPC** *2
Fonction : Déplace l'accumulateur graphique d'un pixel vers le haut.
C. E. :
C. S. : Si le bord est atteint, le sémaphore de retenue est positionné à 1.
R.M. :

Adresse : 0105H
Nom : **TUPC** *2
Fonction : Déplace l'accumulateur graphique d'un pixel vers le haut.
C. E. :
C. S. : Si le bord est atteint, le sémaphore de retenue est positionné à 1.
R.M. :

Adresse : 0108H
Nom : **DOWNC** *2
Fonction : Déplace l'accumulateur graphique d'un pixel vers le bas.
C. E. :
C. S. : Si le bord est atteint, le sémaphore de retenue est positionné à 1.
R.M. :

Adresse : 010BH
Nom : **TDOWNC** *2
Fonction : Déplace l'accumulateur graphique d'un pixel vers le bas.
C. E. :
C. S. : Si le bord est atteint, le sémaphore de retenue est positionné à 1.
R.M. :

Adresse : 010EH
Nom : **SCALXY** *2
Fonction : Etablit l'échelle des coordonnées X Y.
C. E. : BC = X et DE = Y
C. S. : BC = X et DE = Y ajusté (modulo)
R.M. :

Adresse : 0111H
Nom : **MAPXYC** *2
Fonction : Installe les coordonnées de l'écran aux adresses physiques.
C. E. : BC = X et DE = Y
C. S. : HL contient l'adresse en VRAM et A contient le masque à appliquer.
R.M. :
Rem : La routine appelle la ROM étendue si l'écran est en mode multicolore.

Adresse : 0114H
Nom : **FETCHC** *1
Fonction : Lit l'adresse physique courante et le masque (MASK PATTERN).
C. E. : Sans objet
C. S. : HL contient l'adresse physique
R.M. : A, HL

Adresse : 0117H
Nom : **STOREC** *1
Fonction : Sauve l'adresse physique et le masque.
C. E. : HL contient l'adresse physique.
A contient le masque de patron.
C. S. : Sans objet
R.M. : Sans objet

Adresse : 011AH
Nom : **SETATR** *4
Fonction : Positionne les attributs graphiques.
C. E. : A contient le mode courant.
C. S. : ATRBYT (F3F2H) est positionné.
R.M. : Tous

Adresse : 011DH
Nom : **READC** *2
Fonction : Lit l'attribut du pixel courant.
C. E. : SCRMOD contient le mode courant.
CLOC et CMASK (F92AH et F92CH) = pixel courant.
C. S. : A contient l'attribut lu.
R.M. : Tous
Rem : La routine appelle la ROM étendue quand l'écran est en mode multicolore et en mode bitmap.

Adresse : 0120H
Nom : **SETC** *2
Fonction : Place le pixel courant suivant l'attribut spécifié (ATRBYT, adresse F3F2H).
C. E. : ATRBYT, SCRMOD, CLOC et CMASK
C. S. : Rien
R.M. : Tous

Adresse : 0123H
Nom : **NSETCX** *1
Fonction : Remplissage de surface.
C. E. : SCRMOD, ATRBYT, CLOC et CMASK
C. S. : Rien
R.M. : Tous

Adresse : 0126H
Nom : **GTASPC** *1
Fonction : Renvoie le rapport d'aspect des ellipses.
C. E. : Sans objet
C. S. : DE, HL
R.M. : DE, HL

Adresse : 0129H
Nom : **PNTINI** *1
Fonction : Initialisation préalable pour l'instruction PAINT.
C. E. : A contient la couleur, ATRBYT, SCRMOD, CLOC, CMASK.
C. S. : BDRATR (FCB2H)
R.M. : Tous

Adresse : 012CH
Nom : **SCANR** *2
Fonction : Examine les pixels à droite.
C. E. : BDRATR
C. S. : CSAVEA CSAVEM
R.M. : Tous
Rem : La routine appelle la ROM étendue quand l'écran est en mode multicolore et en mode bitmap. Elle utilise une partie de PNTINI.

Adresse : 012FH
Nom : **SCANL** *2
Fonction : Examine les pixels à gauche.
C. E. : BDRATR
C. S. : CSAVEA CSAVEM
R.M. : Tous
Rem : La routine appelle la ROM étendue quand l'écran est en mode multicolore et en mode bitmap.

Adresse : 0132H
Nom : **CHGCAP** *1
Fonction : Allume ou éteint la lampe de la touche CAPS LOCK
C. E. : A contient 0 s'il faut éteindre la lampe, une autre valeur dans le cas contraire.
C. S. : Sans objet
R.M. : AF

Adresse : 0135H
Nom : **CHGSND** *1
Fonction : Change l'état du port sonore à 1 bit (Bit 7 port C).
C. E. : A contient 0 pour positionner le bit à 0, une autre valeur pour positionner le bit à 1.
C. S. : Sans objet
R.M. : AF

Adresse : 0138H
Nom : **RSLREG** *1
Fonction : Lit l'état courant du registre du slot primaire.
C. E. : Sans objet
C. S. : Résultat dans A
R.M. : A

Adresse : 013BH
Nom : **WSLREG** *1
Fonction : Ecrit dans le registre du slot primaire.
C. E. : A contient la valeur à écrire.
C. S. : Sans objet
R.M. : Sans objet

Adresse : 013EH
Nom : **RDVDP** *1
Fonction : Lit le registre d'état du VDP.
C. E. : Sans objet
C. S. : A contient la donnée lue
R.M. : A

Adresse : 0141H
Nom : **SNSMAT** *1
Fonction : Retourne l'état de la ligne spécifiée de la matrice clavier.
C. E. : A contient le numéro de ligne.
C. S. : A contient l'état de la ligne, les bits correspondants aux touches pressées sont mis à 0.
R.M. : A, C

Adresse : 0144H
Nom : **PHYDIO** *1
Fonction : Effectue une opération sur un périphérique de sauvegarde (disques...).
C. E. : Reportez-vous au livre de M. DEVOS sur les disques.
C. S. :
R.M. :
Rem : Dans un configuration minimale, seul un vecteur crochet est fourni.

Adresse : 0147H
Nom : **FORMAT** *1
Fonction : Initialisation d'un périphérique de sauvegarde.
C. E. : ???
C. S. : ???
R.M. : ???
Rem : Dans un configuration minimale, seul un vecteur crochet est fourni.

Adresse : 014AH
Nom : **ISFLIO** *1
Fonction : Teste si on travaille avec un périphérique d'entrée/sortie.
C. E. : Sans objet
C. S. : Si oui, A contient une valeur différente de 0, 0 dans le cas contraire.
R.M. : AF

Adresse : 014DH
Nom : **OUTDLP** *1
Fonction : Sortie vers l'imprimante.
C. E. : A contient le code à imprimer.
C. S. : Sans objet
R.M. : F
Rem : Cette entrée diffère de LPTOUT par plusieurs points :
- les tabulations sont transformées en espaces.
- les hiraganas et les symboles graphiques sont convertis lors de l'utilisation d'une imprimante non-MSX.
- Un saut au message « Device I/O error » est effectué quand l'opération est avortée.

Adresse : 0150H
Nom : **GETVCP** *1
Fonction : Positionnement au début de la file musicale.
C. E. : A contient le numéro de file
C. S. :
R.M. :
Rem : Point d'entrée utilisé uniquement pour jouer de la musique en tâche secondaire.

Adresse : 0153H
Nom : **GETVC2** *1
Fonction : Positionnement précis à l'intérieur d'une file.
C. E. : A contient le numéro de file.
L contient le déplacement dans la file.
C. S. : Rien
R.M. :
Rem : Point d'entrée utilisé uniquement pour jouer de la musique en tâche secondaire.

Adresse : 0156H
Nom : **KILBUF** *1
Fonction : Vide le buffer du clavier
C. E. : Sans objet
C. S. : Sans objet
R.M. : HL

Adresse : 0159H
Nom : **CALBAS** *1
Fonction : Effectue un appel inter-slot dans l'interpréteur Basic (instruction CALL).
C. E. : Adresse dans IX
C. S. : Inconnu
R.M. : Inconnu

Adresse : 015CH
Nom : **SUBROM**
Fonction : Effectue un appel à la ROM secondaire (Sub-ROM)
C. E. : Adresse dans IX (IX est sauvé dans la pile).
C. S. :
R.M. : Le deuxième jeu de registres du Z80 et IY sont réservés.

Adresse : 015FH
Nom : **EXTROM**
Fonction : Effectue un appel à la ROM secondaire (Sub-ROM)
C. E. : Adresse dans IX (IX est sauvé dans la pile).
C. S. :
R.M. : Le deuxième jeu de registres du Z80 et IY sont réservés.

Adresse : 0162H
Nom : **CHKSLZ**
Fonction : Scrute les slots pour trouver la Sub-ROM.
C. E. : Rien
C. S. :
R.M. :

Adresse : 0165H
Nom : **CHKNEW**

Adresse : 0168H
Nom : **EOL**
Fonction : Efface la ligne jusqu'à la fin.
C. E. : H contient le numéro de colonne
L contient le numéro de ligne
Le curseur doit rester inchangé
C. S. : Sans objet
R.M. : Tous

Adresse : 016BH
Nom : **BIGFIL**
Fonction : Identique à FILVRM sauf pour ce qui suit : FILVRM teste si l'écran est en mode 0, 1, 2 ou 3. Dans ce cas, il se comporte comme si le VDP n'avait que 16 Ko de VRAM pour maintenir la compatibilité avec le MSX1. Dans les autres cas, BIGFIL ne teste pas le mode écran et remplit la VRAM comme cela lui a été spécifié par les paramètres.
C. E. : Comme FILVRM
C. S. : Comme FILVRM
R.M. : Comme FILVRM

Adresse : 016EH
Nom : **NSETRD**
Fonction : Prépare le VDP pour la lecture.
C. E. : HL contient l'adresse : tous les bits sont valides.
C. S. : Sans objet
R.M. : AF

Adresse : 0171H
Nom : **NSTWRT**
Fonction : Prépare le VDP pour l'écriture.
C. E. : HL contient l'adresse : tous les bits sont valides.
C. S. : Sans objet
R.M. : AF

Adresse : 0174H
Nom : **NRDVRM**
Fonction : Lit la VRAM adressée par HL : tous les bits sont valides.
C. E. : HL
C. S. : A contient la donnée lue.
R.M. : F

Adresse : 0177H
Nom : **NWRVRM**
Fonction : Ecrit le contenu de A dans la VRAM adressée par HL : tous les bits sont valides.
C. E. : HL, A
C. S. : Sans objet
R.M. : AF

2 ROM étendue MSX2

2.1 Mécanisme d'appel de la ROM étendue

Il existe deux méthodes :

- il faut charger le registre IX avec le point d'entrée (INIPLT) et faire un appel (CALL) à EXTROM (adresse 015FH) :

```
LD   IX, INIPLT
CALL EXTROM
```

- il faut sauvegarder la valeur courante de IX dans la pile, charger IX avec e point d'entrée (INIPLT) et faire un saut (JUMP) à SUBROM (adresse 015CH) :

```
PUSH IX
LD   IX, INIPLT
JP   SUBROM
```

2.2 Routines

Adresse : 0069H
Nom : **PAINT**
Fonction : Peint l'écran graphique.
C. E. :
C. S. :
R. M. :

Adresse : 006DH
Nom : **PSET**
Fonction : Allume un point.
C. E. : Utilise CLOC et CMASK (F92AH et F92CH)
C. S. :
R. M. :

Adresse : 0071H
Nom : **ATRSCN**
Fonction : Détermine l'attribut de couleur.
C. E. : Utilise ATRBYT et SCRMOD
C. S. :
R. M. :

Adresse : 0075H
Nom : **GLINE**
Fonction : Trace une ligne.
C. E. : Utilise GXPOS, GYPOS, GRPACX et GRPACY
C. S. :
R. M. :

Adresse : 0079H
Nom : **DOBOXF**
Fonction : Trace un rectangle plein.
C. E. : Utilise GXPOS, GYPOS, GRPACX et GRPACY
C. S. :
R. M. :

Adresse : 007DH
Nom : **DOLINE**
Fonction : Trace une ligne.
C. E. : Utilise GXPOS, GYPOS, GRPACX et GRPACY
C. S. :
R. M. :

Adresse : 0081H
Nom : **BOXLIN**
Fonction : Dessine un rectangle.
C. E. : Utilise GXPOS, GYPOS, GRPACX et GRPACY
C. S. :
R. M. :

Adresse : 0085H
Nom : **DOGRPH**
Fonction : Dessine une ligne (bas niveau).

C. E. : Utilise GXPOS, GYPOS, GRPACX et GRPACY
C. S. :
R. M. :

Adresse : 0089H
Nom : **GRPPRT**
Fonction : Imprime un caractère sur l'écran graphique.
C. E. : A contient le code à imprimer.
C. S. : Sans objet
R. M. : Sans objet
Rem : Travaille en mode écran 5 à 8.

Adresse : 008DH
Nom : **SCALXY**
Fonction : Détermine l'échelle des coordonnées X et Y
C. E. :
C. S. :
R. M. :
Rem : Travaille en mode écran 5 à 8.

Adresse : 0091H
Nom : **MAPXYC**
Fonction : Transforme les coordonnées en adresses physiques.
C. E. : Utilise CLOC et CMASK.
C. S. :
R. M. :
Rem : Travaille en mode écran 5 à 8.

Adresse : 0095H
Nom : **READC**
Fonction : Lit l'attribut du pixel courant
C. E. : Utilise CLOC, CMASK et ATRBYT.
C. S. :
R. M. :
Rem : Travaille en mode écran 5 à 8.

Adresse : 0099H
Nom : **SETATR**
Fonction : Positionne l'octet d'attribut
C. E. : Utilise ATRBYT
C. S. :
R. M. :
Rem : Travaille en mode écran 5 à 8.

Adresse : 00A1H
Nom : **TRIGHT**
Fonction : Déplace un pixel à droite

C. E. : Utilise CLOC et CMASK
C. S. :
R. M. :
Rem : Travaille en mode multicolore.

Adresse : 00A5H
Nom : **RIGHTC**
Fonction : Déplace un pixel à droite.
C. E. : Utilise CLOC et CMASK
C. S. :
R. M. :
Rem : Travaille en mode multicolore.

Adresse : 00A9H
Nom : **TLEFTC**
Fonction : Déplace un pixel à gauche.
C. E. : Utilise CLOC et CMASK
C. S. :
R. M. :

Adresse : 00ADH
Nom : **LEFTC**
Fonction : Déplace un pixel à gauche.
C. E. : Utilise CLOC et CMASK
C. S. :
R. M. :
Rem : Travaille en mode multicolore.

Adresse : 00B1H
Nom : **TDOWNC**
Fonction : Déplace un pixel vers le bas.
C. E. : Utilise CLOC et CMASK
C. S. :
R. M. :
Rem : Travaille en mode multicolore et en mode écran 5 à 8.

Adresse : 00B5H
Nom : **DOWNC**
Fonction : Déplace un pixel vers le bas.
C. E. : Utilise CLOC et CMASK
C. S. :
R. M. :
Rem : Travaille en mode multicolore.

Adresse : 00B9H
Nom : **TUPC**
Fonction : Déplace un pixel vers le haut.

C. E. : Utilise CLOC et CMASK
C. S. :
R. M. :
Rem : Travaille en mode multicolore et en mode écran 5 à 8.

Adresse : 00BDH
Nom : **UPC**
Fonction : Déplace un pixel vers le haut.
C. E. : Utilise CLOC et CMASK
C. S. :
R. M. :
Rem : Travaille en mode multicolore.

Adresse : 00C1H
Nom : **SCANR**
Fonction : Scrute les pixels sur la droite.
C. E. : Utilise CLOC et CMASK et BDRATR
C. S. :
R. M. :
Rem : Travaille en mode multicolore et en mode écran 5 à 8.

Adresse : 00C5H
Nom : **SCANL**
Fonction : Scrute les pixels sur la gauche.
C. E. : Utilise CLOC et CMASK et BDRATR
C. S. :
R. M. :
Rem : Travaille en mode multicolore et en mode écran 5 à 8.

Adresse : 00C9H
Nom : **NVBXLN**
Fonction : Dessine un rectangle
C. E. : BC et DE contiennent les coordonnées de départ
GXPOS et GYPOS contiennent les coordonnées de fin
ATRBYT contient le code couleur
LOGOPR contient le code logique de l'opération
C. S. : Sans objet
R. M. : Tous

Adresse : 00CDH
Nom : **NVBXFL**
Fonction : Dessine un rectangle
C. E. : BC et DE contiennent les coordonnées de départ
GXPOS et GYPOS contiennent les coordonnées de fin
ATRBYT contient le code couleur
LOGOPR contient le code logique de l'opération
C. S. : Sans objet
R. M. : Tous
Adresse : 00D1H
Nom : **CHGMOD**
Fonction : Place le mode du VDP en accord avec SCRMOD.

C. E. : A contient le mode écran (0 à 8)
C. S. : Sans objet
R. M. : Tous

Adresse : 00D5H
Nom : **INITXT**
Fonction : Initialise l'écran en mode texte (40x24) et active le VDP.
C. E. : TXTNAM, TXTCGP
C. S. : Sans objet
R. M. : Tous

Adresse : 00D9H
Nom : **INIT32**
Fonction : Initialise l'écran en mode texte (32x24) et active le VDP.
C. E. : T32NAM, T32CGP, T32COL, T32ATR, T32PAT
C. S. : Sans objet
R. M. : Tous

Adresse : 00DDH
Nom : **INIGRP**
Fonction : Initialise l'écran en mode haute résolution et active le VDP.
C. E. : GRPNAM, GRPCGP, GRPCOL, GRPATR, GRPPAT
C. S. : Sans objet
R. M. : Tous

Adresse : 00E1H
Nom : **INIMLT**
Fonction : Initialise l'écran en mode multicolore et active le VDP.
C. E. : MLTNAM, MLTCGP, MLTCOL, MLTATR, MLTPAT
C. S. : Sans objet
R. M. : Tous

Adresse : 00E5H
Nom : **SETTXT**
Fonction : Active le VDP pour le mode texte (40x24).
C. E. : TXTNAM, TXTCGP
C. S. : Sans objet
R. M. : Tous

Adresse : 00E9H
Nom : **SETT32**
Fonction : Active le VDP pour le mode texte (32x24).
C. E. : T32NAM, T32CGP, T32COL, T32ATR, T32PAT
C. S. : Sans objet
R. M. : Tous

Adresse : 00EDH
Nom : **SETGRP**
Fonction : Active le VDP pour le mode haute résolution.

C. E. : GRPNAM, GRPCGP, GRPCOL, GRPATR, GRPPAT
C. S. : Sans objet
R. M. : Tous

Adresse : 00F1H
Nom : **SETMLT**
Fonction : Active le VDP pour le mode multicolore.
C. E. : MLTNAM, MLTCGP, MLTCOL, MLTATR, MLTPAT
C. S. : Sans objet
R. M. : Tous

Adresse : 00F5H
Nom : **CLRSPR**
Fonction : Initialise tous les sprites.
Les patrons sont mis à 0.
Les noms des sprites sont pris égaux aux numéros de plans de sprites.
Les couleurs des sprites égales à la couleur d'avant-plan.
Les positions verticales sont mises à 217.
C. E. : SCRMOD
C. S. : Sans objet
R. M. : Tous

Adresse : 00F9H
Nom : **CALPAT**
Fonction : Renvoie l'adresse de la table de patron du sprite.
C. E. : A contient le numéro du sprite
C. S. : HL contient l'adresse de la table
R. M. : AF, DE, HL
Rem: Cette routine est équivalente à celle du Bios sur MSX1.

Adresse : 00FDH
Nom : **CALATR**
Fonction : Renvoie l'adresse de la table de l'attribut du sprite.
C. E. : A contient le numéro du sprite.
C. S. : Hl contient l'adresse de la table.
R. M. : AF, DE, HL
Rem: Cette routine est équivalente à celle du Bios sur MSX1.

Adresse : 0101H
Nom : **GSPSIZE**
Fonction : Renvoie la taille courante du sprite.
C. E. : Sans objet
C. S. : A contient la taille courante (en nombre d'octets)
Le sémaphore de retenue est mis à 1 s'il s'agit d'un sprite 16x16, et 0 dans le cas contraire.
R. M. : AF
Rem : Cette routine est équivalente à celle du Bios sur MSX1.
Adresse : 0105H
Nom : **GETPAT**
Fonction : Renvoie un patron de caractère.

C. E. : A contient le code ASCII du caractère.
C. S. : PATWRK contient le patron du caractère.
R. M. : Tous
Rem : Cette routine est équivalente à celle du Bios sur MSX1.

Adresse : 0109H
Nom : **WRTVRM**
Fonction : Ecrit dans la VRAM adressée par HL
C. E. : HL, A
C. S. : Sans objet
R. M. : AF
Rem : Accepte les adresses allant de 0 à 0FFFFH

Adresse : 010D
Nom : **RDVRM**
Fonction : Lit la VRAM adressée par HL
C. E. : HL
C. S. : A
R. M. : AF
Rem : Accepte les adresses allant de 0 à 0FFFFH

Adresse : 0111H
Nom : **CHGCLR**
Fonction : Change la couleur de l'écran
C. E. : A contient le mode de l'écran
FORCLR contient la couleur d'avant-plan
BAKCLR contient la couleur d'arrière-plan
BDRCLR contient la couleur du bord
C. S. : Sans objet
R. M. : Tous

Adresse : 0115H
Nom : **CLS**
Fonction : Efface l'écran d'affichage.
C. E. : Sans objet
C. S. : Sans objet
R. M. : Tous

Adresse : 0119H
Nom : **CLRTXT**
Fonction : Efface l'écran texte.
C. E. : Sans objet
C. S. : Sans objet
R. M. :

Adresse : 011DH
Nom : **DSPFNK**

Fonction : Affiche le contenu des touches de fonction.
C. E. : Sans objet
C. S. : Sans objet
R. M. : Tous

Adresse : 0121H
Nom : **DELLNO**
Fonction : Efface une ligne en mode texte.
C. E. : Utilise LINLEN, CNSDFG, CRTCNC et FSTPOS
C. S. :
R. M. :

Adresse : 0125H
Nom : **INSLNO**
Fonction : Insère une ligne en mode texte.
C. E. : Utilise LINLEN, CNSDFG, CRTCNC et FSTPOS
C. S. :
R. M. :

Adresse : 0129H
Nom : **PUTVRM**
Fonction : Affiche un caractère sur l'écran texte.
C. E. : Utilise LINLEN, CNSDFG, CRTCNC et FSTPOS
C. S. :
R. M. :

Adresse : 012DH
Nom : **WRTVDP**
Fonction : Ecrit dans un registre du VDP.
C. E. : C contient le numéro de registre
B contient la donnée à écrire
C. S. : Sans objet
R. M. : AF, BC

Adresse : 0131H
Nom : **VDPSTA**
Fonction : Lit un registre d'état du VDP
C. E. : A contient le numéro du registre d'état (0 à 9)
C. S. : A contient l'état du registre sélectionné
R. M. : F

Adresse : 0135H
Nom : **KYKLOK**
Fonction : Ne fait rien
C. E. :
C. S. :
R. M. :

Adresse : 0139H

Nom : **PUTCHR**
Fonction : Ne fait rien
C. E. :
C. S. :
R. M. :

Adresse : 013DH
Nom : **SETPAG**
Fonction : Positionne les registres du VDP pour les changements de page.
C. E. : ACPAGE, DPPAGE
C. S. : Sans objet
R. M. : A

Adresse : 0141H
Nom : **INIPLT**
Fonction : Initialise la palette.
C. E. : Sans objet
C. S. : Sans objet
R. M. : AF, BC, DE
Rem : Chaque palette possède 3 couleurs : Rouge, Vert, Bleu (R, G, B)
Chaque couleur possède 3 bits :

ROUGE	BLEU	VERT
00000RRR	00000BBB	00000GGG

La palette courante est sauvée dans la VRAM.
RSTPLT permet de relire cette palette.

Adresse : 0145H
Nom : **RSTPLT**
Fonction : Lit la palette dans la VRAM.
C. E. : Sans objet
C. S. : Sans objet
R. M. : AF, BC, DE

Adresse : 0149H
Nom : **GETPLT**
Fonction : Lit les codes de couleur dans la palette.
C. E. : A contient le numéro de palette (0 à 15).
C. S. : code ROUGE dans les 4 bits les plus significatifs de A
code BLEU dans les 4 bits les moins significatifs de A
code VERT dans les 4 bits les moins significatifs de E
R. M. : AF, DE

Adresse : 014DH

Nom : **SETPLT**
Fonction : Définit les codes de couleur dans la palette.
C. E. : D contient le numéro de palette
code ROUGE dans les 4 bits les plus significatifs de A
code BLEU dans les 4 bits les moins significatifs de A
code VERT dans les 4 bits les moins significatifs de E
C. S. : Sans objet
R. M. : AF

Adresse : 0151H
Nom : **PUTSPR**
Fonction : Place les sprites
C. E. : Correspond au traitement de l'instruction Basic.
C. S. :
R. M. :

Adresse : 0155H
Nom : **COLOR**
Fonction : Change la couleur de l'écran
Change la couleur des sprites.
Change la palette.
C. E. : Correspond au traitement de l'instruction Basic.
C. S. :
R. M. :

Adresse : 0159H
Nom : **SCREEN**
Fonction : Change le mode écran
C. E. : Correspond au traitement de l'instruction Basic.
C. S. :
R. M. :

Adresse : 015DH
Nom : **WIDTHS**
Fonction : Change la largeur de l'écran texte.
C. E. : Correspond au traitement de l'instruction Basic.
C. S. :
R. M. :

Adresse : 0161H
Nom : **VDP**
Fonction : Positionne un registre du VDP.
C. E. : Correspond au traitement de l'instruction Basic.
C. S. :
R. M. :

Adresse : 0165H

Nom : **VDPF**
Fonction : Lit le registre courant du VDP.
C. E. : Correspond au traitement de l'instruction Basic.
C. S. :
R. M. :

Adresse : 0169H
Nom : **BASE**
Fonction : Positionne les registres de base du VDP.
C. E. : Correspond au traitement de l'instruction Basic.
C. S. :
R. M. :

Adresse : 016DH
Nom : **BASEF**
Fonction : Lit les registres de base du VDP.
C. E. : Correspond au traitement de l'instruction Basic.
C. S. :
R. M. :

Adresse : 0171H
Nom : **VPOKE**
Fonction : Ecrit un octet dans la VRAM.
C. E. : Correspond au traitement de l'instruction Basic.
C. S. :
R. M. :

Adresse : 0175H
Nom : **VPEEK**
Fonction : Lit un octet dans la VRAM.
C. E. : Correspond au traitement de l'instruction Basic.
C. S. :
R. M. :

Adresse : 0179H
Nom : **SETS**
Fonction : Positionne le ton 'BEEP'.
C. E. : Correspond au traitement de l'instruction Basic.
C. S. :
R. M. :

Adresse : 017DH
Nom : **BEEP**
Fonction : Envoie un son au haut-parleur.
C. E. : Sans objet
C. S. : Sans objet
R. M. : Tous

Adresse : 0181H

Nom : **PROMPT**
Fonction : Affiche l'indicatif du système.
C. E. : Sans objet
C. S. : Sans objet
R. M. : Tous

Adresse : 0185H
Nom : **SDFSCR**
Fonction : Initialisation mode écran 4 à 7.
C. E. :
C. S. :
R. M. :

Adresse : 0189H
Nom : **SETSCR**
Fonction : Initialisation mode écran 4 à 7.
C. E. :
C. S. :
R. M. :

Adresse : 018DH
Nom : **SCOPY**
Fonction : Copie la VRAM, les tableaux et les fichiers disque.
C. E. :
C. S. :
R. M. :

Adresse : 0191H
Nom : **BLTVV**
Fonction : Copie la VRAM dans la VRAM (copie logique).
C. E. :
C. S. :
R. M. :

Adresse : 0195H
Nom : **BLTVM**
Fonction : Copie un tableau dans la VRAM (copie rapide).
C. E. :
C. S. :
R. M. :

Adresse : 0199H
Nom :
Fonction :
C. E. :
C. S. :
R. M. :

Adresse : 019DH

Nom :
Fonction :
C. E. :
C. S. :
R. M. :

Adresse : 01A1H
Nom :
Fonction :
C. E. :
C. S. :
R. M. :

Adresse : 01A5H
Nom :
Fonction :
C. E. :
C. S. :
R. M. :

Adresse : 01A9H
Nom :
Fonction :
C. E. :
C. S. :
R. M. :

Adresse : 01ADH
Nom : **NEWPAD**
Fonction : Gestion de la fonction PADDLE étendue.
C. E. : A contient le code du test

8	Etalonne le crayon lumineux (255 si valide).
9	Renvoie la coordonnée X.
10	Renvoie la coordonnée Y.
11	Renvoie l'état du commutateur du crayon (255 si pressé).
12	Etalonne la connexion d'une souris au port 1.
13	Renvoie la position X de la souris.
14	Renvoie la position Y de la souris.
15	(Toujours 0).
16	Etalonne la connexion d'une souris au port 2.
17	Renvoie la position X de la souris.
18	Renvoie la position Y de la souris.
19	(Toujours 0).

C. S. : A
R. M. : Tous

Adresse : 01B1H

Nom : **GETPUT**
Fonction : Lit l'heure et la date.
C. E. :
C. S. :
R. M. :

Adresse : 01B5H
Nom : **CHGMDP**
Fonction : Met le VDP en concordance avec SCRMOD.
La palette est initialisée.
C. E. : A contient le mode écran (0 à 8).
C. S. : Sans objet
R. M. : Tous

Adresse : 01B9H
Nom : **RESV1**
Fonction : Réservé. Non utilisé.
C. E. :
C. S. :
R. M. :

Adresse : 01BDH
Nom : **KNJPRT**
Fonction : Imprime un caractère Kanji sur l'écran graphique (5 à 8).
C. E. : BC = code du caractère Kanji.
A = Mode d'affichage.
C. S. : Sans objet
R. M. : AF

Adresse : 01F5H
Nom : **REDCLK**
Fonction : Lit l'heure sur l'horloge interne.
C. E. : C contient l'adresse de la RAM du circuit d'horloge.
bit - 7 6 5 4 3 2 1 0
C = X X M1 M0 A3 A2 A1 A0
C. S. : A = données lues (4 bits moins significatifs).
R. M. : F

Adresse : 01F9H
Nom : **WRTCLK**
Fonction : Ecrit une donnée dans l'horloge interne.
C. E. : C contient l'adresse de la RAM de l'horloge.
A contient la donnée à écrire
bit - 7 6 5 4 3 2 1 0
C = X X M1 M0 A3 A2 A1 A0
C. S. : Sans objet
R. M. : F

3 Paramètres de la région de communication

La région de communication commence par quelques courtes routines qui améliorent la lecture/écriture inter-slot.

PPI = 0A8H (port A du PPI)

Adresse	Long	Nom	Fonction
F380	5	RDPRIM	Lecture d'un slot dans E
F385	7	WRPRIM	Ecriture dans un slot
F38C	14	CLRPRIM	Appel d'un slot
F39A	20	USRTAB	Table des adresses définies par l'instruction DEFUSRn. Elle occupe 10*2 octets.
F3AE	1	LINL40	Longueur de ligne en mode SCREEN 0 (40). Vaut 39 à l'initialisation.
F3AF	1	LINL32	Longueur de ligne en mode SCREEN1 (32). Vaut 31 à l'initialisation.
F3B0	1	LINLEN	Longueur de ligne courante (WIDTH)
F3B1	1	CRTCNC	Compteur de ligne (longueur de page). Vaut 24 en standard.
F3B2	1	CLMSLT	Nombre de caractères pour le TAB. Vaut 14 en standard.
F3B3	2	TXTNAM	Adresse de la TNP en mode 0
F3B5	2	TXTCOL	Adresse de la TC en mode 0
F3B7	2	TXTCGP	Adresse de la TGP en mode 0
F3B9	2	TXTATR	Adresse de la TAS en mode 0
F3BB	2	TXTPAT	Adresse de la TGS en mode 0
F3BD	2	T32NAM	Adresse de la TNP en mode 1
F3BF	2	T32COL	Adresse de la TC en mode 1
F3C1	2	T32CGP	Adresse de la TGP en mode 1
F3C3	2	T32ATR	Adresse de la TAS en mode 1
F3C5	2	T32PAT	Adresse de la TGS en mode 1
F3C7	2	GRPNAM	Adresse de la TNP en mode 2
F3C9	2	GRPCOL	Adresse de la TC en mode 2
F3CB	2	GRPCGP	Adresse de la TGP en mode 2
F3CD	2	GRPATR	Adresse de la TAS en mode 2
F3CF	2	GRPPAT	Adresse de la TGS en mode 2
F3D1	2	MLTNAM	Adresse de la TNP en mode 3
F3D3	2	MLTCOL	Adresse de la TC en mode 3
F3D5	2	MLTCGP	Adresse de la TGP en mode 3
F3D7	2	MLTATR	Adresse de la TAS en mode 3
F3D9	2	MLTPAT	Adresse de la TGS en mode 3
F3DB	1	CLIKSW	0 = pas de click (1 = click)
F3DC	1	CSRY	Position courante curseur Y
F3DD	1	CSRX	Position courante curseur X
F3DE	1	CNSDFG	Affichage des touches de fonction 0 = pas d'affichage/ 1 = affichage
F3DF	1	RG0SAV	Contenu de R0 (VDP)
F3E0	1	RG1SAV	Contenu de R1

F3E1	1	RG2SAV	Contenu de R2		
F3E2	1	RG3SAV	Contenu de R3		
F3E3	1	RG4SAV	Contenu de R4		
F3E4	1	RG5SAV	Contenu de R5		
F3E5	1	RG6SAV	Contenu de R6		
F3E6	1	RG7SAV	Contenu de R7		
F3E7	1	STATFL	0		
F3E8	1	TRGFLG	FF		
F3E9	1	FORCLR	Couleur d'avant-plan (15 par défaut)		
F3EA	1	BAKCLR	Couleur d'arrière-plan (4 par défaut)		
F3EB	1	BDRCLR	Couleur de bord (4 par défaut)		
F3EC	3	MAXUPD	JMP	$CODE C3 00 00	
F3EF	3	MINUPD	JMP	$CODE C3 00 00	
F3F2	1	ATRBYT	Octet d'attribut (15)		
F3F3	2	QUEUES	Adresse des tables de files utilisées par QUEUTL		
F3F5	1	FRCNEW	255		
F3F6	1	SNCCNT	Intervalle pour scruter les touches (1)		
F3F7	1	REPCNT	Répétition (50)		
F3F8	2	PUTPNT	Adresse de l'octet courant à écrire dans le tampon du clavier.		
F3FA	2	GETPNT	Adresse de l'octet courant à lire dans le tampon du clavier.		
F3FC	10	CS120	Paramètres cassette		
		5 valeurs pour vitesse = 1200 bauds			
		LOW01	DB	83	largeur état bas du 0
		HIGH01	DB	92	largeur état haut du 0
		LOW11	DB	38	largeur état bas du 1
		HIGH11	DB	45	largeur état haut du 1
			DB	HEDLEN*2/256	
			HEDLEN = 2000		longueur des bits de l'en-tête pour un en-tête court.
		5 valeurs pour vitesse = 2400 bauds			
		LOW02	DB	37	largeur état bas du 0
		HIGH02	DB	45	largeur état haut du 0
		LOW12	DB	14	largeur état bas du 1
		HIGH12	DB	22	largeur état haut du 1
			DB	HEDLEN*4/256	
F406	2	LOW	1200 bauds par défaut		
F408	2	HIGH			
F40A	1	HEADER	DB	HEDLEN*2/256	
F40B	2	ASPCT1	DW1	$CODE+256	256/aspect
F40D	2	ASPCT2	DW2	$CODE+256	256*aspect
F40F	5	ENDPRG	DB	« ; »	Indicateur de fin de programme (RESUME NEXT)
F414	1	ERRFLG	Utilisé pour sauver le numéro d'erreur		
F415	1	LPTPOS	Position de la tête de l'imprimante (0)		
F416	1	PRTFLG	Sémaphore imprimante (0 = IMPRIMANTE)		
F417	1	NTMSXP	Différent de 0 si imprimante non MSX		
F418	1	RAWPRT	Différent de 0 si l'impression est en mode graphique (RAW MODE)		
F419	1	VLZADR	Adresse du caractère pris en compte par VAL		
F41B	1	VLZDAT	Caractère remplacé par 0 par VAL		

F41C	2	CURLIN	Numéro courant de la ligne en cours d'exécution
F41F	KBFLEN	KBUF	Tampon pour le codage d'une ligne Basic
F55D	1	BUFMIN	Virgule utilisée par 'INPUT' (préchargée ou en ROM) pour que le pointeur vers les données démarre sur une virgule ou un signe de fin
F55E	BUFLEN+3	BUF	Zone tampon pour le clavier
F660	1	ENDBUF	Fin de tampon pour les longues lignes
F661	1	TTYPOS	Charge la position du terminal
F662	1	DIMFLG	Sémaphore de l'instruction DIM
F663	1	VALTYP	Indicateur du type de variable
F664	0	OPRTYP	Type d'opérateur
F664	1	DORES	Selon que « crunch » peut ou ne peut pas intégrer les mots résidents dans les 8 Ko au moment d'explorer les DATAs, les chaînes non mises entre guillemets ne pourront être « crunchées »
F665	1	DONUM	Sémaphore pour le cas où « crunch » = 0
F666	2	CONTXT	Adresse du caractère courant dans le texte
F668	1	CONSAV	Mot-clé sauvé (code instruction) après que CHRGET a été appelé
F669	1	CONTYP	Sauve le type de valeur de la constante
F66A	8	CONLO	Sauve la valeur de la constante
F672	2	MEMSIZ	Valeur supérieure de la mémoire utilisable par le Basic. Cette valeur est modifiée par l'instruction CLEAR
F674	2	STKTOP	Adresse supérieure du pointeur de pile (SP)
F676	2	TXTTAB	Pointeur vers le début du texte. Ne doit plus être changé après avoir été initialisé par INIT.
F678	2	TEMPPT	Pointeur vers le premier descripteur de TEMP libre, initialisé pour pointer sur TEMPST.
F67A	3*NUMTMP	TEMPST	Chargement pour les descripteurs de NUMTMP TEMP.
F698	3	DSCTMP	Employé durant l'utilisation des fonctions alphanumériques.
F69B	2	FRETOP	Sommet de l'espace libre pour les chaînes
F69D	2	TEMP3	Charge l'adresse de fin d'une zone de chaîne dans 'GARBAGE COLLECTION' et momentanément utilisé par FRMEVL
F69F	2	TEMP8	Utilisé par 'GARBAGE COLLECTION'
F6A1	2	ENDFOR	Pointeur de texte sauvé à la fin de l'instruction FOR
F6A3	2	DATLIN	Adresse de la ligne de DATA
F6A5	1	SUBFLG	Sémaphore pour FOR et USR
F6A6	1	FLGINP	Sémaphore pour INPUT et READ
F6A7	2	TEMP	Code temporaire pour certaines instructions
F6A9	1	PTRFLG	Vaut 0 si aucun pointeur vers un numéro de ligne n'a été converti. Différent de 0 si ces pointeurs existent.
F6AA	1	AUTFLG	Sémaphore : 0 = AUTO : 1 = PAS AUTO
F6AB	2	AUTLIN	Numéro de la ligne courante inséré par AUTO
F6AD	2	AUTINC	Valeur de l'incrément de AUTO
F6AF	2	SAVTXT	Pointeur pour l'instruction RESUME
F6B1	2	SAVSTK	Sauve l'adresse de la pile avant de manipuler une erreur qui changerait cette adresse de pile
F6B3	2	ERRLIN	Contient le numéro de la ligne de la dernière erreur rencontrée
F6B5	2	DOT	Garde le numéro de la ligne courante pour EDIT et

			LIST
F6B7	2	ERRTXT	Pointeur vers le texte pour l'instruction RESUME
F6B9	2	ONELIN	Numéro de la ligne pour un traitement d'erreur
F6BB	1	ONEFLG	Vaut 1 pendant l'exécution d'une routine de traitement d'erreur
F6BC	2	TEMP2	Formule pour l'évaluation de TEMP
F6BE	2	OLDLIN	Ancien numéro de ligne, activé par ^C, STOP ou END dans un programme
F6C0	2	OLDTXT	Adresse du dernier octet exécuté
F6C2	2	VARTAB	Adresse de la table des variables simples
F6C4	2	ARYTAB	Pointeur vers le début de la table des tableaux. Incrémenté de 6 quand une nouvelle variable simple est ajoutée et initialisée par CLEARC
F6C6	2	STREND	Fin de la zone utilisée. Augmenté quand une nouvelle zone ou une variable simple est rencontrée par CLEARC
F6C8	2	DATPTR	Pointe vers DATA. Initialisé pour pointer vers 0 au début de TXTTAB par RESTORE quand il est appelé par CLEARC. Remis à jour par l'exécution d'un READ.
F6CA	26	DEFTBL	Donne le type de valeur par défaut pour chaque lettre de l'alphabet. Utilisé par CLEAR et changé par DEFSTR, DEFINT, DEFSGN, DEFDBL et utilisé par PTRGET quand aucun signe ! # % ou $ ne suit un nom de variable.
F6E4	2	PRMSTK	Définition prévue pour le bloc sur la pile
F6E6	2	PRMLEN	Nombre d'octets dans la table active
F6E8	PRMSIZ	PARM1	Table de définition des paramètres actifs
F74C	2	PRMPRV	Initialise PRMSTK
F74E	2	PRMLN2	Table des paramètres du bloc à construire
F750	PRMSIZ	PARM2	Place pour garder les paramètres devant être utilisés par PTRGET
F7B5	2	ARYTA2	Point d'arrêt pour une simple recherche
F7B7	1	NOFUNS	Vaut 0 si aucune fonction n'est active
F7B8	2	TEMP9	GARBAGE COLLECTION TEMP doit être chaîné par l'intermédiaire de blocs paramétrés
F7BA	2	FUNACT	Compte les fonctions actives
F7BC	8	SWPTMP	Valeur de la première variable SWAP chargée
F7C4	1	TRCFLG	0 signifie que TRACE n'est pas utilisé
F7C5	43	FBUFFR	TAMPON pour FOUT
F7F0	2	DECTMP	Utilisé pour conversion entier vers flottant
F7F2	2	DECTM2	Utilisé pour les divisions
F7F4	1	DECCNT	Utilisé pour les divisions
F7F6	16	DAC	Accumulateur mathématique
F806	48	HOLD8	; 80*X
F836	8	HOLD2	; 2*X
F83E	8	HOLD	; 1*X
F847	16	ARG	Accumulateur secondaire
F857	8	RNDX	Contient le dernier nombre 'Random' généré
F85F	1	MAXFIL	Nombre maximum de fichiers
F860	2	FILTAB	Pointeur vers l'adresse des données d'un fichier
F862	2	NULBUF	Pointeur vers le tampon du fichier 0
F864	2	PTRFIL	Pointeur vers les données d'un fichier sélectionné

F866	0	RUNFLG	Différent de 0 s'il doit tourner après chargement
F866	11	FILNAM	Nom du fichier et extension
F871	11	FILNM2	Autres noms de fichiers pour NAME
F87C	1	NLONLY	Différent de 0 si un programme est en cours de chargement
F87D	2	SAVEND	Fin du sauvetage d'une image binaire ou mémoire
F87F	16*10	FNKSTR	Contenu des touches de fonction (F1 à F10)
F91F	3	CGPNT	Adresse du premier patron de caractères dans la ROM
F922	2	NAMBAS	Adresse de la TNP courante
F924	2	CGPBAS	Adresse de la TGP courante
F926	2	PATBAS	Adresse de la TGS courante
F928	2	ATRBAS	Adresse de la TAS courante
F92A	2	CLOC	
F92C	1	CMASK	
F92D	2	MINDEL	
F92F	2	MAXDEL	
F931	2	ASPECT	Etat de l'aspect du cercle
F933	2	CENCNT	Fin des calculs cercle
F935	1	CLINEF	Sémaphore pour dessiner une ligne vers le centre
F936	2	CNPNTS	Pointe vers le point à tracer
F938	1	CPLOTF	Sémaphore de polarité des points à tracer
F939	2	CPCNT	1/8 du nombre de points dans le cercle
F93B	2	CPCNT8	Nombre de points dans le cercle
F93D	2	CRCSUM	Calcul du cercle
F93F	2	CSTCNT	Début du calcul
F941	1	CSCLXY	Mise à l'échelle des points X et Y
F942	2	CSAVEA	Sauve la zone ADVGRP
F944	1	CSAVEM	Sauve la zone ADVGRP
F945	2	CXOFF	Sauve la distance de X par rapport au centre du cercle
F947	2	CYOFF	Sauve la distance de Y par rapport au centre du cercle
F949	1	LOHMSK	Zone de sauvetage dans la RAM pour le débordement à gauche
F94A	1	LOHDIR	
F94B	2	LOHADR	
F94D	2	LOHCNT	
F94F	2	SKPCNT	Comptage des sauts
F951	2	MOVCNT	Comptage des mouvements
F953	1	PDIREC	Direction pour la peinture
F954	1	LFPROG	
F955	1	RTPROG	
F956	2	MCLTAB	
F958	1	MCLFLG	Sémaphore pour indiquer PLAY ou DRAW
F959	24	QUETAB	4 files (6 octets chacune)
F971	4	QUEBAK	Pour BCKQ
F975	MUSQLN	VOICAQ	File de la voie A
F9F5	MUSQLN	VOICBQ	File de la voie B
FA75	MUSQLN	VOICCQ	File de la voie C
FAF5	RSIQLN	RS2IQ	File d'entrée de la RS232

FB35	1	PRSCNT	D1-D0 = # chaînes analysées
			D7 = 0 si c'est la première analyse, 1 dans le cas contraire

FB36	2	SAVSP	Sauve le pointeur vers la pile principale pendant

			l'exécution de PLAY
FB38	1	VOICEN	Active la voie courante qui vient d'être analysée
FB39	2	SAVVOL	Sauve le volume pour un arrêt
FB3B	1	MCLLEN	
FB3C	2	MCLPTR	
FB3E	1	QUEUEN	
FB3F	1	MUSICF	Sémaphore d'interruption musicale
FB40	1	PLYCNT	# instructions PLAY mises en file pour des tâches d'arrière-plan
FB41	VCBSIZ	VCBA	Data résident pour la voie 0
FB66	VCBSIZ	VCBB	Data résident pour la voie 1
FB8B	VCBSIZ	VCBC	Data résident pour la voie 2
FBB0	1	ENSTOP	Différent de 0 si un démarrage à chaud est possible
FBB1	1	BASROM	Différent de 0 si le Basic est en ROM
FBB2	24	LINTTB	Table des fins de lignes
FBCA	2	FSTPOS	Première position quand INLIN est entré
FBCC	1	CURSAV	Code sauvant la zone du curseur
FBCD	1	FNKSWI	Indique la touche de fonction affichée
FBCE	10	FNKFLG	Indique la touche assignée à un éventuel périphérique
FBD8	1	ONGSBF	Sémaphore général évènement
FBD9	1	CLIKFL	
FBDA	11	OLDKEY	Etat de l'ancienne touche clavier
FBE5	11	NEWKEY	Etat de la nouvelle touche clavier
FBF0	40	KEYBUF	Tampon des codes de touches clavier
FC18	0	BUFEND	Fin de KEYBUF
FC18	40	LINWRK	Zone de 'scratch' pour le gestionnaire d'écran
FC40	8	FATWRK	Zone de 'scratch' pour le convertisseur de patron
FC48	2	BOTTOM	Début de la mémoire RAM
FC4A	2	HIMEM	Fin de la mémoire RAM
FC4C	3*NUMPTR	TRPTBL	Table des pièges
FC9A	1	RTYCNT	
FC9B	1	INTFLG	Sémaphore d'interruption
FC9C	1	PADY	Valeur Y de la manette analogique
FC9D	1	PADX	Valeur X de la manette analogique
FC9E	2	JIFFY	
FCA0	2	INTVAL	Valeur de l'intervalle pour ON INTERVAL GOSUB
FCA2	2	INTCNT	Compteur de l'intervalle
FCA4	1	LOWLIM	Utilisé pendant la lecture d'une cassette
FCA5	1	WINWID	Utilisé pendant la lecture d'une cassette
FCA6	1	GRPHED	Sémaphore pour la sortie d'un caractère graphique
FCA7	1	ESCCNT	Compteur pour la séquence ESCAPE
FCA8	1	INSFLG	Sémaphore du mode insertion
FCA9	1	CSRSW	Switch d'affichage du curseur (ON/OFF)
FCAA	1	CSTYLE	Caractère du curseur
FCAB	1	CAPST	Etat de CAPS LOCK
FCAC	1	KANAST	Etat du verrou des kanas
FCAD	1	KANAMD	Différent de 0 si JIS
FCAE	1	FLBMEM	0 si un programme Basic est en cours de chargement
FCAF	1	SCRMOD	Mode courant de l'écran
FCB0	1	OLDSCR	Zone de sauvetage de l'ancien mode écran
FCB1	1	CASPRV	Sauvetage du caractère précédent (CAS)
FCB2	1	BRDATR	Couleur de bord de l'écran pour PAINT

FCB3	2	GXPOS	Position pixel graphique en X
FCB5	2	GYPOS	Position pixel graphique en Y
FCB7	2	GRPACX	Accumulateur graphique X
FCB9	2	GRPACY	Accumulateur graphique Y
FCBB	1	DRWFLG	Etiquette pour DRAW
FCBC	1	DRWSCL	Facteur d'échelle pour DRAW : 0 = pas d'échelle
FCBD	1	DRWANG	Angle pour DRAW (0-3)
FCBE	1	RUNBNF	Sémaphore pour BLOAD, BSAVE
FCBF	2	SAVENT	Adresse de départ pour BSAVE
FCC1	4	EXPTBL	Table des sémaphores des slots étendus. Contient 255 s'il est étendu
FCC5	4	SLTTBL	Etat courant pour chaque registre du slot étendu
FCC9	64	SLTATR	Adresses attribuées à chaque slot
FD09	128	SLTWRK	Zone de travail pour chaque slot
FD89	16	PROCNM	Nom CALL terminé par 0
FD99	1	DEVICE	ID pour une cartouche (0 .. 3)

4 Table des vecteurs crochets

Voici pour terminer ce chapitre, la table des vecteurs crochets. Chaque entrée de la table se compose de 5 octets. En standard, dans un système sans périphérique, ces 5 octets ont la valeur C9H (RET). Si on utilise des extensions (disque, RS232...), la plupart de ces vecteurs sont interceptés et ils contiennent alors une instruction de déroulement C3 (JUMP) suivie d'une adresse de transfert. Les 2 octets restant (5 octets) permettent d'effectuer un saut inter-slot éventuel.

Adresse	Nom	Fonction
FD9A	HOKJMP	Début de la zone des crochets (HOOK)
FD9A	H.KEYI	Dans MSXIO au début du traitement des interruptions. réalise un traitement additionnel d'interruption comme par exemple la RS232C
FD9F	H.TIMI	Dans MSXIO, au traitement de l'interruption du Timer. permet un autre traitement d'interruption invoqué par le Timer
FDA4	H.CHPU	Dans MSXIO, au début de la routine CHPUT (écriture d'un caractère sur l'écran). Permet en sortie (output) l'utilisation d'autres périphériques que la console
FDA9	H.DSPC	Dans MSXIO, au début de la routine DSPCSR (affichage du curseur)
FDAE	H.ERAC	Dans MSXIO, au début de la routine ERACSR (effacement du curseur)
FDB3	H.DSPF	Dans MSXIO, au début de la routine DSPFNK (affichage des touches de fonction). Permet en sortie (output) l'utilisation d'autres périphériques que la console
FDB8	H.ERAF	Dans MSXIO, au début de la routine ERAFNK (effacement des touches de fonction). Permet en sortie (output) l'utilisation d'autres périphériques que la console
FDBD	H.TOTE	Dans MSXIO, au début de la routine TOTEXT (force l'écran en mode texte). Permet en sortie (output) l'utilisation d'autres périphériques que la console
FDC2	H.CHGE	Dans MSXIO, au début de la routine CHGET (lecture d'un caractère). Permet en sortie (output) l'utilisation d'autres périphériques que la console
FDC7	H.INIP	Dans MSXIO, au début de la routine INIPAT (initialisation d'un

		patron). Permet l'utilisation d'autres tables de caractères.
FDCC	H.KEYC	Dans MSXIO, au début de la routine KEYCOD (codeur des touches). Permet l'utilisation d'autres attributions pour les touches clavier
FDD1	H.KYEA	Dans MSXIO, au début de la routine KYEASY. Permet l'utilisation d'autres attributions pour les touches clavier
FDD6	H.NMI	Dans MSXIO, au début de la routine NMI (interruption non masquable). Permet le traitement de NMI
FDDB	H.PINL	Dans MSXINL, au début de la routine PINLIN (programme d'entrée des lignes). En entrée (input), permet l'utilisation d'autres périphériques que la console ou une autre manière d'entrer des lignes.
FDE0	H.QINL	Dans MSXINL, au début de la routine QINLIN (impression de la ligne en 'Input' et du ?). En entrée (input), permet l'utilisation d'autres périphériques que la console ou une autre manière d'entrer des lignes.
FDE5	H.INLI	Dans MSXINL, au début de la routine INLIN (Line Input). En entrée (input), permet l'utilisation d'autres périphériques que la console ou une autre manière d'entrer des lignes.
FDEA	H.ONGO	Dans MSXSTS, au début de la routine ONGOTOP (traitement de l'instruction On Goto). Permet l'utilisation d'autres périphériques d'interruption.
FDEF	H.DSKO	Dans MSXSTS, au début de la routine DSKO$ (sortie en lecture du disque). Pour installer un driver de disques
FDF4	H.SETS	Dans MSXSTS, au début de la routine SETS (attributs de l'instruction SET). Pour installer un driver de disques
FDF9	H.NAME	Dans MSXSTS, au début de la routine NAME (Instruction NAME). Pour installer un driver de disques
FDFE	H.KILL	Dans MSXSTS, au début de la routine KILL (instruction KILL). Pour installer un driver de disques
FE03	H.IPL	Dans MSXSTS, au début de la routine IPL (charge le programme initial). Pour installer un driver de disques
FE08	H.COPY	Dans MSXSTS, au début de la routine COPY (copie de fichiers). Pour installer un driver de disques
FE0D	H.CMD	Dans MSXSTS, au début de la routine CMD (instruction COMMAND). Pour installer un driver de disques
FE12	H.DSKF	Dans MSXSTS, au début de la routine DSKF (Instruction 'Disk Free' : place disponible sur un disque). Pour installer un driver de disques
FE17	H.DSKI	Dans MSXSTS, au début de la routine DSKI (introduction de données sur un disque). Pour installer un driver de disques
FE1C	H.ATTR	Dans MSXSTS, au début de la routine ATTR$ (Instruction ATTR$). Pour installer un driver de disques
FE21	H.LSET	Dans MSXSTS, au début de la routine LSET (alignement à gauche dans un fichier). Pour installer un driver de disques
FE26	H.RSET	Dans MSXSTS, au début de la routine RSET (alignement à droite dans un fichier). Pour installer un driver de disques
FE2B	H.FIEL	Dans MSXSTS, au début de la routine FIELD. Pour installer un driver de disques
FE30	H.MKI$	Dans MSXSTS, au début de la routine MKI$ (transformation d'un entier en une chaîne de caractères). Pour installer un driver de disques
FE35	H.MKS$	Dans MSXSTS, au début de la routine MKS$ (transformation d'une expression simple précision en chaîne de caractères). Pour installer un driver de disques
FE3A	H.MKD$	Dans MSXSTS, au début de la routine MKD$ (transformation d'une expression double précision en chaîne de caractères). Pour installer

		un driver de disques
FE3F	H.CVI	Dans MSXSTS, au début de la routine CVI (transformation d'une chaîne en nombre entier). Pour installer un driver de disques
FE44	H.CVS	Dans MSXSTS, au début de la routine CVS (transformation d'une chaîne en nombre simple précision). Pour installer un driver de disques
FE49	H.CVD	Dans MSXSTS, au début de la routine CVD (transformation d'une chaîne en nombre double précision). Pour installer un driver de disques
FE4E	H.GETP	Dans SPCDSK, à la routine GETPTR (saisit le pointeur de fichier). Pour installer un driver de disques
FE53	H.SETF	Dans SPCDSK, à la routine SETFIL (active le pointeur vers le fichier). Pour installer un driver de disques
FE58	H.NOFO	Dans SPCDSK, à la routine NOFOR (pas de clause FOR). Pour installer un driver de disques
FE5D	H.NULO	Dans SPCDSK, à la routine NULOPN (aucun fichier ouvert). Pour installer un driver de disques
FE62	H.NTFL	Dans SPCDSK, à la routine NTFLO (aucun fichier de numéro 0). Pour installer un driver de disques
FE67	H.MERG	Dans SPCDSK, à la routine MERGE (ajoute le texte d'un fichier sur disque au fichier qui existe déjà). Pour installer un driver de disques
FE6C	H.SAVE	Dans SPCDSK, à la routine SAVE. Pour installer un driver de disques
FE71	H.BINS	Dans SPCDSK, à la routine BINSAV (sauve un fichier binaire). Pour installer un driver de disques
FE76	H.BINL	Dans SPCDSK, à la routine BINLOD (charge un fichier binaire). Pour installer un driver de disques
FE7B	H.FILE	Dans SPCDSK, à la commande FILES. Pour installer un driver de disques
FE80	H.DGET	Dans SPCDSK, à la routine DGET (permet de lire sur le disque). Pour installer un driver de disques
FE85	H.FILO	Dans SPCDSK, à la routine FILOU1 (sortie de fichier 1). Pour installer un driver de disques
FE8A	H.INDS	Dans SPCDSK, à la routine INDSKC (place un caractère sur le disque). Pour installer un driver de disques
FE8F	H.RSLF	Dans SPCDSK, pour sélectionner à nouveau un « ancien » disque. Pour installer un driver de disques
FE94	H.SAVD	Dans SPCDSK, pour sauver le disque courant. Pour installer un driver de disques
FE99	H.LOC	Dans SPCDSK, à la fonction LOC. Pour installer un driver de disques
FE9E	H.LOF	Dans SPCDSK, à la fonction LOF. Pour installer un driver de disques
FEA3	H.EOF	Dans SPCDSK, à la fonction EOF. Pour installer un driver de disques
FEAB	H.FPOS	Dans SPCDSK, à la fonction FPOS. Pour installer un driver de disques
FEAD	H.BAKU	Dans SPCDSK, à la routine BACKUP. Pour installer un driver de disques
FEB2	H.PARD	Dans SPCDEV, à la routine PARDEV (analyse du nom de device). Pour ajouter un périphérique logique
FEB7	H.NODE	Dans SPCDEV, à la routine NODEVN (pas de nom de périphérique). Pour installer, par défaut, un autre périphérique
FEBC	H.POSD	Dans SPCDEV, à la routine POSDSK (disque possible). Pour installer un driver de disques
FEC1	H.DEVN	Dans SPCDEV, à la routine DEVNAM (nom de périphérique). Pour

		ajouter un périphérique logique
FEC6	H.GEND	Dans SPCDEV, au GENDSP (contrôleur général des périphériques). Pour ajouter un périphérique logique
FECB	H.RUNC	Dans BIMISC, à la routine RUNC ('RUN' et exécute un CLEAR).
FED0	H.CLEA	Dans BIMISC, à la routine CLEARC (initialise la table des variables)
FED5	H.LOPD	Dans BIMISC, à la routine LOPDFT (boucle la table des variables et les vides). Pour utiliser d'autres valeurs par défaut pour variables.
FEDA	H.STKE	Dans BIMISC, à la routine STKERR (erreur de pile)
FEDE	H.ISFL	Dans BIMISC, à la routine ISFLIO (détecte l'existence d'un fichier en entrée/sortie)
FEE4	H.OUTD	Dans BIO, à la routine OUTDO (sort un caractère sur écran ou sur imprimante)
FEE9	H.CRDO	Dans BIO, à la routine CRDO (exécute un retour chariot)
FEEE	H.DSKC	Dans BIO, à la routine à la routine DSKCHI (place un caractère sur le disque)
FEF3	H.DOGR	Dans GENGRP, à la routine DOGRPH (exécute une fonction graphique)
FEF8	H.PRGE	Dans BINTRP, à la routine PRGEND (fin de programme)
FEFD	H.ERRP	Dans BINTRP, à la routine ERRPRT (imprime un message d'erreur)
FF02	H.ERRF	Dans BINTRP. A la fin de l'impression d'un message d'erreur
FF07	H.READ	Dans BINTRP, à l'impression du message OK
FF0C	H.MAIN	Dans BINTRP, à l'entrée MAIN
FF11	H.DIRD	Dans BINTRP, à l'entrée DIRDO (exécute une instruction en mode direct)
FF16	H.FINI	Dans BINTRP, à la fin de l'interprétation d'une instruction
FF1B	H.FINE	Dans BINTRP, à la fin de l'interprétation
FF20	H.CRUN	Dans BINTRP, à l'entrée du CRUNCHER
FF25	H.CRUS	Dans BINTRP, au début de la recherche d'une instruction dans la table alphabétique
FF2A	H.ISRE	Dans BINTRP, lors de la découverte d'un mot réservé dans la table en phase CRUNCH
FF2F	H.NTFN	Dans BINTRP, lorsque le mot réservé est suivi d'un numéro de ligne (GOTO, GOSUB...)
FF34	H.NOTR	Dans BINTRP, lorsque le mot n'est pas réservé
FF39	H.SNGF	Dans BINTRP, pour installer un autre package mathématique
FF3E	H.NEWS	Dans BINTRP, au début d'une nouvelle instruction
FF43	H.GONE	Dans BINTRP, lors d'une instruction de déroutement (GOTO...)
FF48	H.CHRG	Dans BINTRP, lors de la saisie d'un caractère
FF4D	H.RETU	Dans BINTRP, lors du traitement d'un RETURN
FF52	H.RMB	Dans BINTRP, lors de l'instruction PRINT
FF57	H.RMB	Dans BINTRP, dans le corps du traitement du PRINT
FF5C	H.FINP	Dans BINTRP, à la fin du traitement du PRINT
FF61	H.TRMN	Dans BINTRP, lors du traitement d'un DATA ou d'un INPUT
FF66	H.FRME	Dans BINTRP, lors de l'évaluation d'une formule
FF6B	H.NTPL	Dans BINTRP, pour installer un autre package mathématique
FF70	H.EVAL	Dans BINTRP, lors de l'évaluation d'une expression
FF75	H.OKNO	Dans BINTRP, lors de l'évaluation d'une expression transcendante (SIN, COS...)
FF7A	H.FING	Dans BINTRP, à la fin de l'évaluation d'une expression transcendante
FF7F	H.ISMI	Dans BINTRP, à la routine ISMID$ (traitement de MID$)
FF84	H.WIDTH	Dans BINTRP, à la routine WIDTHS (nombre de caractères par ligne)
FF89	H.LIST	Dans BINTRP, à la routine LIST (Donne la 'liste' d'un programme)

FF8E	H.BUFL	Dans BINTRP, à la routine BUFLIN (tampon de ligne)
FF93	H.FRQI	Dans BINTRP, à la routine FRQINT
FF98	H.SCNE	Dans BINTRP, lors de la conversion d'un numéro de ligne en pointeur et inversement
FF9D	H.FRET	Dans BISTRS, à la routine FRETMP (places temporairement vides)
FFA2	H.PTRG	Dans BIPTRG, à la routine PTRGET (lit la position du pointeur). Pour utiliser d'autres noms de variables que les noms par défaut
FFA7	H.PHYD	Dans MSXIO, à la routine PHYDIO (entrée/sortie physique du disque)
FFAC	H.FORM	Dans MSXIO, à la routine FORMAT (permet de formater un disque)
FFB1	H.ERRO	Dans BINTRP, à la routine ERROR (traitement des erreurs). Pour dépister les erreurs dans les programmes d'applications
FFB6	H.LPTO	Dans MSXIO, à la routine LPTOUT (sortie de l'imprimante). Pour utiliser une imprimante (autre que celle par défaut)
FFBB	H.LPTS	Dans MSXIO, à la routine LPTSTT (état de l'imprimante).Pour utiliser une imprimante (autre que celle par défaut)
FFC0	H.SCRE	Dans MSXSTS, à l'entrée de l'instruction SCREEN
FFC5	H.PLAY	Dans MSXSTS, à l'entrée de l'instruction PLAY. Pour utiliser l'instruction PLAY
FFCA	ENDWRK	Fin de la zone de travail

CHAPITRE 6

STRUCTURE DE LA MÉMOIRE

1 Généralités

Le microprocesseur Z80 qui équipe les ordinateurs MSX dispose d'un espace d'adressage de 64 Ko. Cela signifie qu'il est capable d'adresser n'importe quel octet d'une mémoire de 65536 positions tant dans le but de lire que d'écrire cet octet.

Un problème survient lorsque l'ensemble des espaces mémoire, nécessaires pour un programme donné, dépasse 64 Ko. Ainsi, pour un MSX2 de 64 Ko par exemple, il y a :

- une ROM de 16 Ko contenant le Bios
- une ROM de 16 Ko contenant le Basic MSX1
- une ROM de 16 Ko contenant le Basic MSX2
- une RAM de 64 Ko

TOTAL : 112 Ko

Le Z80 ne pouvant adresser que 64 Ko, les concepteurs du système MSX ont donc dû trouver un solution à ce problème.

2 Description de la page et du slot

L'espace d'adressage de 64 Ko du Z80 a été partagé en 4 pages :

	HEXADECIMAL	DECIMAL
PAGE 0 :	0000 à 3FFF	00000 à 16383
PAGE 1 :	4000 à 7FFF	16384 à 32767
PAGE2 :	8000 à BFFF	32768 à 49151
PAGE 3 :	C000 à FFFF	49152 à 65535

On peut donc dire qu'à tout moment le Z80 « voit » 4 pages. Quand il regarde (quand il adresse pour être plus technique) la page 0, il y voit 16384 octets toujours placés aux adresses 0 à 3FFF. Quand il regarde la page 1, il y voit 16384 octets toujours placés aux adresses 4000 à 7FFF.

L'astuce utilisée par les concepteurs du système consiste à permettre au microprocesseur de choisir une page 0 parmi un stock de plusieurs pages 0. De même le microprocesseur peut choisir une page 1 parmi un stock de plusieurs pages 1 et ainsi de suite pour la page 2 et la page 3.

Quand le Z80 a choisi pour page 0 celle de son stock de pages 1 que nous appellerons X, il est évident qu'il peut lire ou écrire dans n'importe laquelle des 16384 positions de cette page X, mais d'autre part, il est tout aussi évident qu'il ne peut ni lire ni écrire dans les autres pages 0 du stock puisqu'il ne peut en choisir qu'UNE parmi ce stock.

Les différentes pages constituant le stock on été placées dans ce que la norme MSX appelle des SLOTS. Un slot peut contenir 4 pages, à savoir une page 0, une page 1, une page 2 et une page 3. Donc on peut dire que la capacité mémoire d'un slot est de 64 Ko. La norme MSX ayant prévu un maximum de 4 slots par machine, la capacité totale d'adressage de notre MSX est : 4 x 65536 = 262 144 octets, réparties en 4 pages 0, 4 pages 1, 4 pages 2 et 4 pages 3. Littéralement, le mot anglais slot signifie fente. En effet, le système de découpage de la mémoire maximum en slots a été prévu pour accroître la mémoire MSX par l'adjonction de cartouches en mémoire. Comment ajouter ces cartouches d'extension sinon en les glissant dans une fente découpée dans le boîtier de votre MSX et au fond de laquelle il y a un connecteur pour relier cette cartouche d'extension au microprocesseur.

C'est ici qu'il faut cesser d'appeler les slots par leur traduction française fente. En effet une fente implique la présence d'un connecteur en son fond tandis que le terme slot est employé indistinctement pour un slot muni d'un connecteur ou non.

Ainsi, la norme MSX a défini que chaque MSX doit posséder un slot interne qui est appelé slot 0. Ce slot n'a pas de connecteur et contient toujours la ROM Bios comme page 0 et la ROM Basic comme page 1.

Les slots 1 et 2 sont habituellement utilisés pour les cartouches d'extension bien que rien dans la norme MSX n'impose que ce soit les slots 1 et 2 qui soient réservés à cet effet. Il existe par exemple des MSX où ce sont les slots 1 et 3 qui sont munis des connecteurs pour les cartouches d'extension.

En ce qui concerne la mémoire RAM, la norme MSX n'a pas imposé de slot précis, elle peut être placée au gré des constructeurs dans n'importe quel slot sauf bien entendu dans les slots réservés aux cartouches d'extension. Certains constructeurs là encore la placent dans le slot 3, d'autres dans le slot 1 et d'autres encore la partagent entre le slot 0 et le slot 3. De même, il n'est pas obligatoire qu'il y ait toujours deux slots munis de connecteurs ni qu'il y ait 4 slots systématiquement dans chaque MSX.

3 Les différents types de slots

3.1 Les slots primaires

La norme MSX a prévu un maximum de 4 slots qu'on appelle slots primaires. Les fentes munies de connecteurs pour cartouches de jeux ou d'extension sont en général des slots primaires. De même, un MSX de base a toujours un slot primaire 0 qui est situé à l'intérieur du boîtier de votre ordinateur et de 1 à 3 slots primaires munis d'un connecteur pour cartouches d'extension. Il en résulte que si votre MSX dispose de deux fentes, vous pourrez y glisser deux cartouches d'extension mémoire de 64 Ko, ce qui équivaut à étendre la mémoire de votre MSX de 128 Ko.

Comment faire maintenant pour indiquer au microprocesseur qu'il doit regarder (ou adresser) tel slot comme page 0, tel autre slot comme page 1, tel autre comme page 2, et finalement tel autre comme page 3 ? Les MSX ont été munis d'un registre de sélection de slots. Il s'agit du SLOT SELECT REGISTER présent au port d'entrée/sortie A8 (hexadécimal) ou 168 (décimal).

BIT 8	SS2	PAGE 3
BIT 7	SS1	
BIT 6	SS2	PAGE 2
BIT 5	SS1	
BIT 4	SS2	PAGE 1
BIT 3	SS1	
BIT 2	SS2	PAGE 0
BIT 1	SS1	

SS2	SS1	
0	0	SLOT 0
0	1	SLOT 1
1	0	SLOT 2
1	1	SLOT 3

En Basic, il n'est pas du tout conseillé de programmer le registre de slot car le Basic a toujours besoin des pages 0 et 1 du slot 0 pour les ROMs Bios et Basic, et la page 3 ne peut pas être commutée cas elle contient la région de communication du Basic. Seule la page 2 peut éventuellement être commutée mais comme il est interdit de commuter une page dans laquelle figurent les instructions de commutation (on ne scie pas la branche sur laquelle on est assis...), on est pratiquement obligé de passer par une routine en langage machine que l'on placera dans la page 3.

Par contre, il est aisé d'obtenir le contenu de ce registre pour connaître dans quel slot se trouve la mémoire RAM employée par le Basic. En effet, celle-ci est toujours située dans les pages 2 et 3 et il suffit dès lors de décortiques le contenu des bits 5 à 8 du registre de sélection de slot pour le savoir.

Exemple :

```
10 A = INP(&HA8)
20 PRINT RIGHT$(« 00000000 » + BIN$(A), 8
```

3.2 Les slots secondaires

Supposons que vous disposiez d'un MSX1 avec 2 ports d'extension (c'est-à-dire 2 slots avec connecteur) et que vous désiriez y connecter un disque via un contrôleur qui s'enfichera dans l'un des deux ports d'extension et une interface RS232 (permet de relier 2 ordinateurs entre eux) qui se glissera dans l'autre port d'extension.

Vos 2 ports d'extension sont maintenant occupés et il est dès lors apparemment devenu impossible d'y connecter une cartouche de 64 Ko de RAM supplémentaire. D'ailleurs, le slot 0 est occupé par les ROMs Bios et Basic, et le dernier slot est probablement occupé par la mémoire RAM interne de votre MSX.

Détrompez-vous, il est encore possible d'ajouter des slots supplémentaires à cet MSX. En effet, la norme MSX a prévu qu'un slot primaire peut être étendu à 4 slots secondaires.

Concrètement, dans notre exemple, nous allons enlever, d'un des deux connecteurs, l'interface qui y est glissée, nous y placerons en lieu et place un câble spécial au bout duquel se trouve une interface munie de 4 connecteurs. Ce type d'interface s'appelle un SLOT EXPANSION INTERFACE c'est-à-dire une interface d'extension de slots.

Dans deux des 4 connecteurs, nous pourrons replacer l'interface que nous avions enlevé et ajouter une cartouche d'extension de mémoire de 64 Ko et le tour est joué...

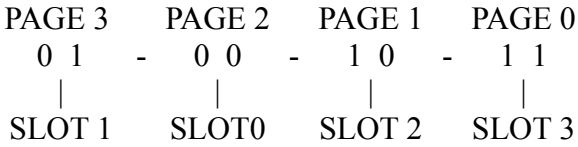
Plus théoriquement, chaque slot primaire peut être étendu à 4 slots secondaires. Cela nous donne donc un total de 16 slots secondaires. Un slot secondaire peut contenir 4 pages de 16 Ko soit 64 Ko exactement comme un slot primaire. La capacité maximum d'un MSX est donc devenue : 16 slots de 64 Ko soit 1 048 576 octets soit 1 Mo. Il n'y a pas de différence entre la mémoire présente en slot primaire ou en slot secondaire, le système les reconnaît l'une comme l'autre.

Il n'est pas non plus obligatoire qu'un MSX ayant un slot primaire étendu à 4 slots secondaires ait tous ses slots primaires étendus. Il peut très bien n'y en avoir qu'un. De même, il n'est pas nécessaire que les 4 slots secondaires soient tout utilisés. Enfin, les constructeurs peuvent étendre un slot primaire interne à la console en 4 slots secondaires, internes. Dans ce cas, il n'y a pas de connecteur extérieur, les slots secondaires internes étant alors utilisés pour ajouter de la mémoire ou pour intégrer un contrôleur disque ou encore une interface RS232 ou des programmes propres à certains constructeurs et tout cela directement dans le boîtier de votre console.

La sélection d'un slot secondaire se fait relativement à un slot primaire. Il faut donc sélectionner le slot primaire contenant le slot secondaire choisi avant de pouvoir sélectionner le slot secondaire choisi. C'est pourquoi on utilise la notation suivante pour parler d'un des 16 slots secondaires pouvant exister : SLOT 3, 2.

Le chiffre 3 placé avant la virgule indique le slot primaire et le chiffre 2 indique le slot secondaire de ce slot primaire 3.

Voyons maintenant le principe de la sélection d'une case ou page de 16 Ko. Vous vous rappelez que le registre de sélection des slots, présent au port A8H, permettait de déterminer dans quel slot le microprocesseur doit regarder pour trouver une page donnée. Ainsi, si nous mettons la valeur 4BH dans ce registre, nous aurons les bits suivants :



Ce qui signifie que la page 0 se trouvera dans le slot primaire 3, la page 1 dans le slot 2, la page 2

dans le slot 0 et la page 3 dans le slot 1.

Il faudra maintenant pour CHAQUE SLOT PRIMAIRE désigner quel SLOT SECONDAIRE va être sélectionné. Cela se fera grâce à quatre REGISTRES DE SÉLECTION DE SLOT SECONDAIRE.

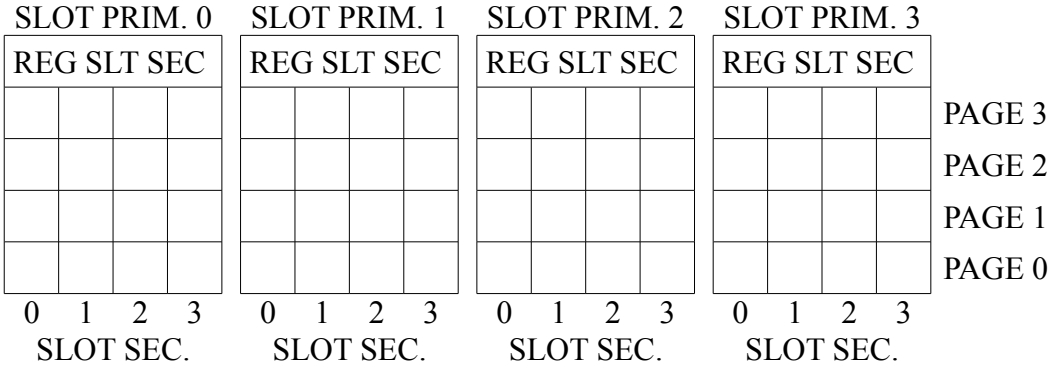
Un système MSX contient 4 slots primaires, donc il y a 1 registre de sélection de slot primaire. Il y a 4 slots primaires partagés en slots secondaires, donc il doit y avoir 4 registres de sélection de slot secondaire.

Pour faciliter la sélection, les concepteurs du système MSX ont décidé de placer le registre de sélection de slot secondaire dans la position mémoire FFFFH de chacun des slots primaires, c'est-à-dire la dernière position mémoire du slot.

Comme chacun des 4 slots primaires dispose d'une position FFFFH, nous avons bien 4 registres de sélection. D'autre part, il faut tenir compte que la page 3 de chaque slot primaire étendu ne dispose plus que de 16383 octets au lieu de 16384 puisque la dernière position de cette page 3 est réservée pour le registre de sélection de slot secondaire.

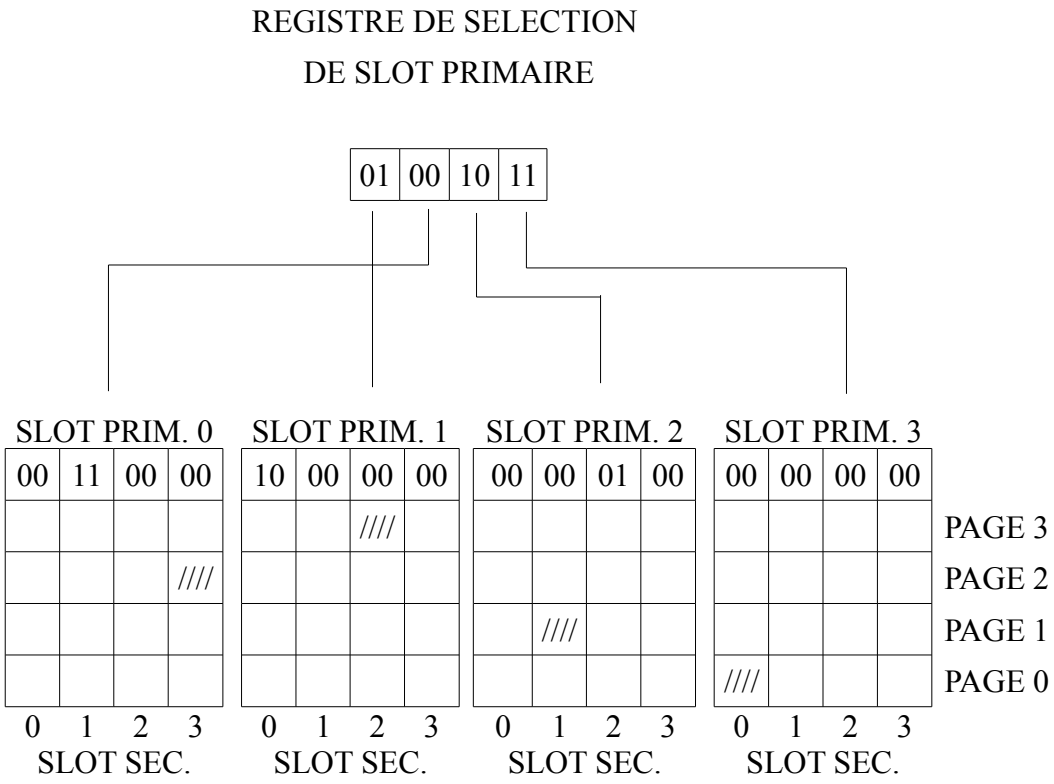
Il faut aussi remarquer que dans ce système la position FFFFH n'est plus une position RAM normale, mais est devenue un registre de sélection. De ce fait, il était nécessaire de fournir un moyen de tester si la position FFFFH est une position RAM normale reflétant un slot primaire non étendu en slots secondaires ou un registre de sélection de slot secondaire. Le système suivant a été choisi :

On écrit une valeur quelconque dans la position FFFFH, puis on relit cette position. Si la valeur lue est identique à celle écrite, c'est qu'il s'agit d'une mémoire RAM. Si la valeur lue est l'inverse (au niveau des bits) de la valeur écrite, alors c'est qu'il s'agit bien d'un registre de sélection de slot secondaire. Avant d'aller plus loin, voici le diagramme montrant la présence de ces registres de sélection.



Les 4 registres sont constitués de 8 bits et le format utilisé est exactement le même que celui du registre de sélection de slot primaire.

L'exemple suivant vous donne la programmation des registres de sélection pour les cases hachurées.



En conclusion, n'oubliez pas qu'il s'agit d'une sélection en cascade. En effet, le registre de sélection de slot primaire sélectionne d'abord les slots primaires pour chaque page et ensuite, les registres de sélection de slot secondaire définissent le slot secondaire choisi. Concrètement, si la page 2 est choisie dans le slot primaire 1, les registres de sélection de slot secondaire des autres slots primaires (0, 1, 2à peuvent indiquer n'importe quelle valeur pour la page 2 sans que cela ait un quelconque effet.

3.3 Le MEMORY MAPPER MSX2

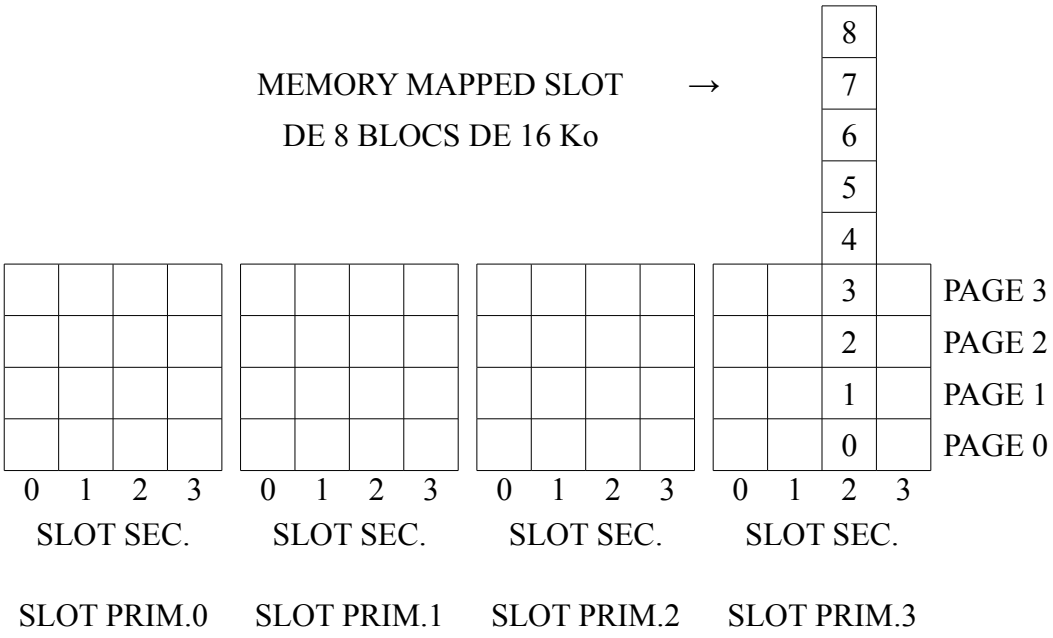
Les MSX2 peuvent être munis d'un nouveau dispositif permettant de porter la taille mémoire maximum d'un slot de 64 Ko à 4 méga-octets. La norme MSX2 a, en effet, défini qu'un unique slot quelconque (primaire non-étendu ou secondaire) pouvait être configuré en MEMORY MAPPED SLOT.

Le « Memory mapped slot » (MMS) est un slot contenant de 8 blocs (128 Ko) à 256 blocs (4 Mo) de 16 Ko de mémoire RAM. Le nombre exact de ces blocs de 16 Ko dépend du constructeur.

Cependant, comme il s'agit d'un slot, notre MSX devra sélectionner ce slot de façon classique c'est-à-dire en programmant le registre de sélection de slot primaire et éventuellement le registre de sélection de slot secondaire si le « Memory mapped slot » se trouve en slot secondaire.

De ce fait, la règle générale, qui dit que le Z80 ne peut voir que 64 Ko (4 pages de 16 Ko) simultanément, s'applique aussi au « Memory mapped slot ». Il faudra donc avoir un système qui permette de choisir pour chacune des 4 pages classiques (page 0 à page 3) quel bloc de 16 Ko sera choisi dans cette réserve de 256 blocs au maximum. Pour ce faire, les blocs de 16 Ko présents dans la « Mapped memory » seront numérotés de 0 à 255 (de 0 à 7 si elle ne contient que 8 blocs). Voici le nouveau diagramme de représentation de la mémoire si le « Memory mapped slot » se trouve

dans le slot primaire 3, slot secondaire 2 et contient 8 blocs de 16 Ko :



Pour allouer un des 256 blocs de 16 Ko à la page 0, il faudra placer dans un registre spécial le numéro du bloc de 16 Ko choisi. De même, pour allouer un bloc à la page 1 ou 2 ou 3, il faudra placer le numéro de bloc de 16 Ko choisi dans un registre spécialement réservé à cette page 1 ou 2 ou 3.

Il y a un registre de sélection par page soit un total de 4 registres. Ces registres ont été implantés dans les adresses de ports suivantes :

PAGE 0	PORT ADDRESS FCH
PAGE 1	PORT ADDRESS FDH
PAGE 2	PORT ADDRESS FEH
PAGE 3	PORT ADDRESS FFH

Si nous voulons faire voir au microprocesseur le bloc 0 en tant que page 0, le bloc 1 en tant que page 1, le bloc 2 en tant que page 2 et le bloc 3 en tant que page 3 comme le diagramme ci-dessus, nous devons d'abord programmer le registre de sélection de slot primaire avec la valeur FFH, ce qui signifie que toutes les pages sont dans le slot primaire 3, puis nous devons programmer le registre de sélection de slot secondaire avec la valeur AAH pour indiquer que toutes les pages se trouvent dans le slot secondaire 2 et enfin écrire dans les registres du Mapped slot les valeurs suivantes :

Registre FC :	valeur 00	Page 0 = bloc 0
Registre FD :	valeur 01	Page 1 = bloc 1
Registre FE :	valeur 02	Page 2 = bloc 2
Registre FF :	valeur 03	Page 3 = bloc 3
Registre A8 :	valeur FF	Page 0 à 3 dans SP 3
Registre FFFF :	valeur AA	Page 0 à 3 dans SS2

Bien entendu il est possible d'allouer à une page quelconque n'importe quel bloc de 0 à 255. Par défaut, à l'allumage du système, nous trouvons curieusement la programmation suivante :

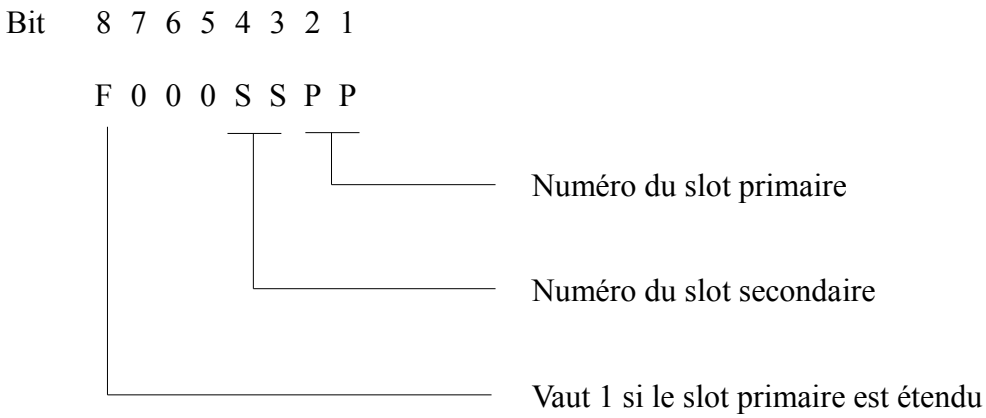
Registre FC :	valeur 03	Page 0 = bloc 3
Registre FD :	valeur 02	Page 1 = bloc 2
Registre FE :	valeur 01	Page 2 = bloc 1
Registre FF :	valeur 00	Page 3 = bloc 0

La programmation a été intentionnellement réalisée pour que la page 3 - qui est toujours nécessaire en Basic pour la région de communication - occupe le bloc 0/ Cela permettra le chargement consécutif des blocs 1 à X par un chargeur adapté puisque les blocs 1 à X forment une série continue de blocs libres. La programmation en langage machine de ces registres se fait simplement par les instructions suivantes :

OUT	(0FC), A	pour écrire le registre
IN	A, (0FC)	pour lire le registre

4 Les tables des slots en mémoire

Lors de l'allumage de votre MSX en Disk-basic ou en MSX-DOS, le système recherche la présence de RAM dans le but de constituer une table indiquant pour chaque page où se trouve cette RAM. Cette recherche se limite à trouver de la RAM dans les 4 pages sans poursuivre ses investigations pour vérifier s'il n'y a pas plusieurs slots contenant de la mémoire RAM aux mêmes adresses de page. La méthode utilisée pour indiquer où se trouve la RAM est identique au SLOT SELECT CHARACTER utilisé par les routines de commutation de lots. Vous trouverez donc dans la table des pages RAM un caractère par page dont la signification est la suivante :



La table des pages est implantée en F341 à F344.

F341 : adresse de la RAM pour la page 0
F341 : adresse de la RAM pour la page 0
F341 : adresse de la RAM pour la page 0
F341 : adresse de la RAM pour la page 0

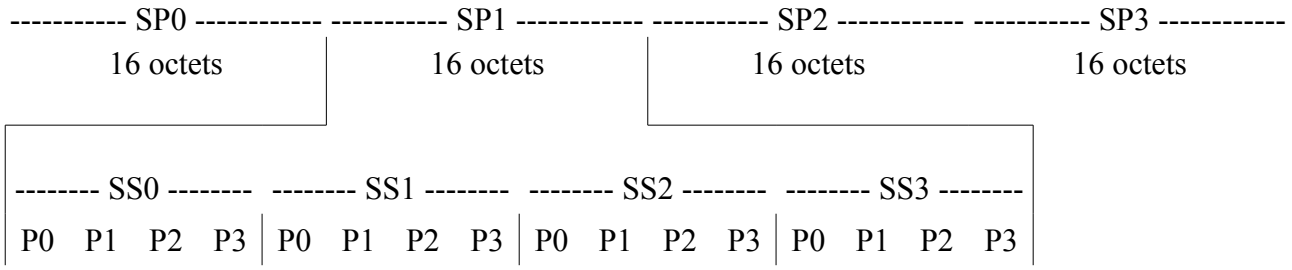
Il existe une autre table définissant pour votre MSX le type de chaque slot primaire. Le type peut être normal ou étendu. Il est ainsi facile de savoir si un slot primaire donné est étendu à 4 slots secondaires. Cette table est implantée aux adresses FCC1 à FCC4. On y trouve 0 quand un slot primaire n'est pas étendu ou 80H s'il est étendu.

FCC1 : type du slot primaire 0
FCC2 : type du slot primaire 1
FCC3 : type du slot primaire 2
FCC4 : type du slot primaire 3

D'autre part, la valeur du registre de sélection de slot secondaire est maintenue pour chaque slot primaire dans une table située en FCC5. Vous y trouverez directement le contenu du registre sans devoir aller lire la position FFFF du slot primaire concerné et sans devoir inverser les bits.

FCC5 : Contenu du registre de sélection de slot secondaire du slot primaire 0
FCC6 : Contenu du registre de sélection de slot secondaire du slot primaire 1
FCC7 : Contenu du registre de sélection de slot secondaire du slot primaire 2
FCC8 : Contenu du registre de sélection de slot secondaire du slot primaire 3

Il existe finalement une dernière table qui va indiquer pour chaque page de chaque sous-slot ce que contient la ROM qui y serait éventuellement placée. Cette table est constituée d'un octet par ROM de 16 Ko. Sa structure est la suivante :



La table est composée de 64 octets soit 16 octets par slot primaire (SP0 à SP1). Ces 16 octets sont eux-mêmes partagés en 4 octets par slot secondaire (SS0 à SS1). Chacun des 4 octets d'un slot secondaire représente les 4 pages de ce slot (P0 à P3).

Le contenu de chaque position (P0 à p3) a la signification suivante :

Bit	8	7	6	5	4	3	2	1
	B	D	I	0	0	0	0	0

B est 1 si la ROM contient un programme Basic.

D est 1 si la ROM contient des devices supplémentaires.

I est 1 si la ROM contient un programme en langage machine.

5 Les routines de manipulation des slots

Il existe 5 routines de manipulation des slots. Ces routines ont le gros avantage d'avoir été reproduites deux fois dans le système. Elles sont, en effet, présente dans la ROM Bios utilisée par le Basic et dans la RAM lorsqu'on travaille en MSX-DOS. Un autre avantage est qu'elles ont été implantées aux mêmes adresses tant dans la ROM que dans la RAM.

Parmi ces 5 routines, on en trouve 2 qui permettent de lire et d'écrire un octet de/vers n'importe quel slot, 2 autres permettent d'appeler une sous-routine présente dans un autre slot et finalement la dernière permet de valider un slot de façon permanente.

SLOT SELECT CHARACTER

Bit	8	7	6	5	4	3	2	1
	F	0	0	0	S	S	P	P

PP indique le slot primaire 0 à 3 en 2 bits.

SS indique le slot secondaire 0 à 3 en 2 bits si le bit F est à l'état 1.

F indique, lorsqu'il est à 1, qu'il faut tenir compte des bits 3 et 4. Autrement dit, le slot primaire est étendu.

EXEMPLE : 8BH = 139 décimal = 1 0 0 0 1 0 1 1 indique qu'il s'agit du slot 3, 2 soit le slot primaire 3 - slot secondaire 2.

5.1 RDSLT

ADRESSE : 000C ReaD SLoT

FONCTION : La routine READ SLOT (lecture du slot) permet d'obtenir dans le registre A un octet présent à l'adresse donnée par le registre HL et dont le slot est indiqué par le registre A.

ENTREE : A doit indiquer le Slot select character.

HL doit indiquer l'adresse à lire.

SORTIE : A donne le contenu de la mémoire à l'adresse HL

NOTE : Une instruction DI (Disable Interrupt) est automatiquement exécutée par la routine mais vous pouvez (et en principe devez) ré-autoriser les interruptions par une instruction EI (Enable Interrupt).

PRESERVE : Le registre HL est préservé.

5.2 WRSLT

ADRESSE : 0014 WRite SLoT

FONCTION : Permet d'écrire le caractère présent dans le registre E dans la position RAM dont l'adresse est donnée par le registre HL et le Slot select character par le registre A.

ENTREE : A Slot select character.

HL Adresse de la position RAM à écrire.

E Code à écrire en mémoire.

SORTIE : Rien.

NOTE : Une instruction DI (Disable Interrupt) est automatiquement exécutée par la routine

mais vous pouvez (et en principe devez) ré-autoriser les interruptions par une instruction EI (Enable Interrupt).

PRESERVE : Les registres E et HL sont préservés.

5.3 CALSLT

ADRESSE : 001C CALl SLoT

FONCTION : C'est probablement l'appel de fonction le plus intéressant car il permet d'appeler une routine qui se trouve dans n'importe quelle page de n'importe quel slot. Ainsi, bien que le travail en MSX-DOS nécessite les 4 pages de RAM, il sera possible d'appeler n'importe quelle routine présente dans la ROM Bios ou dans la Sub-ROM du MSX2. Le MSX-DOS s'enrichit donc grâce à cet appel de toutes les fonctions normales du MSX qu'elles soient graphiques, sonores ou autres.

Comme la plupart des routines du Bios nécessite le passage de paramètres par les registres A, BC, DE ou HL, cet appel va utiliser deux registres (IX et IY) habituellement non utilisés par les routines du Bios.

La partie haute du registre IY devra contenir le Slot select character du slot où se trouve la routine et le registre IX l'adresse de la routine que l'on désire appeler dans ce slot.

Il est bon de noter ici que la position RAM FCC1 donne le Slot select character de la ROM Bios et la position FAF8 celui de la ROM d'extension du MSX2 (Sub-ROM).

ENTREE : IY doit contenir le Slot select character de la ROM ou de la RAM contenant la routine à appeler.

IX doit contenir l'adresse de la routine.

SORTIE : Cet appel ne produit aucun résultat en soi, mais la routine appelée retourne peut-être des valeurs pour les registres A, BC, DE ou HL.

NOTE : Une instruction DI (Disable Interrupt) est automatiquement exécutée par la routine mais vous pouvez (et en principe devez) ré-autoriser les interruptions par une instruction EI (Enable Interrupt).

PRESERVE : Les registres A, BC, DE et HL sont préservés par cet appel, mais répétons encor eque la routine appelée peut modifier ces registres.

5.4 ENASLT

ADRESSE : 0024 ENAble SLoT

FONCTION : L'appel ENASLT permet de valider de façon permanente la page indiquée par le registre H et qui est présente dans le slot A.

ENTREE : A doit contenir le Slot select character.

H doit contenir les 2 chiffres hexadécimaux supérieurs de l'adresse de la page choisie (00 pour la page 0, 40 pour la page 1, 80 pour la page 2 et C0 pour la page 3).

SORTIE : Rien.

NOTE : Une instruction DI (Disable Interrupt) est automatiquement exécutée par la routine

mais vous pouvez (et en principe devez) ré-autoriser les interruptions par une instruction EI (Enable Interrupt).

D'autre part, si votre appel s'adresse au Bios ROM, vous pouvez sélectionner n'importe quelle page sauf la page 0 car la routine de commutation y réside.

Par contre, si vous employez l'appel du MSX-DOS, vous pouvez commuter la page 0 mais pas la page 3 pour la même raison que ci-dessus.

PRESERVE : Aucun registre n'est préservé.

5.5 CALLF

ADRESSE : 0030 CALL Far

FONCTION : Cet appel permet d'appeler (CALL) une routine présente dans un slot fixe et connu d'avance. Elle a l'avantage sur CALSLT d'être plus concise, mais présente l'inconvénient suivant : il faut connaître le slot où se trouve la routine appelée au moment de l'assemblage du programme.

L'adresse où cet appel est implanté (0030) peut être appelé par une instruction RST 30 en un seul octet ce qui est plus concis que le traditionnel CALL 0030 qui en demande 3.

ENTREE : La technique utilisée pour appeler une routine dans un autre slot consiste à placer dans la source de votre programme les 3 lignes suivantes :

```
RST   30      ; appel de CALLF.
DEFB  yy      ; placer ici le Slot select character.
DEFW  xxxx    ; placer ici l'adresse de la routine que vous désirez appeler.
```

SORTIE : Rien par l'appel en soi. Voyez la routine appelée pour connaître les résultats sortis.

PRESERVE : L'appel en soir ne modifie aucun registre, mais la routine appelée peut modifier n'importe quel registre. Voir la routine appelée.

6 Procédure d'initialisation

Le Basic MSX cherche d'abord la RAM disponible de l'adresse 0BFFFH à l'adresse 8000H (incluant les premières dans les slots secondaires) puis active la page contenant la plus grande RAM. Si plusieurs pages sont semblables, il sélectionne la page de gauche (voir le dessin ci-dessus).

Il cherche ensuite la RAM disponible de l'adresse 0FFFFH à l'adresse 0C000H et travaille de la même manière que ci-dessus. Finalement, MSX-Basic cherche un bloc continu de RAM de l'adresse 0FFFFH à l'adresse 8000H et y installe le sommet de la table des variables.

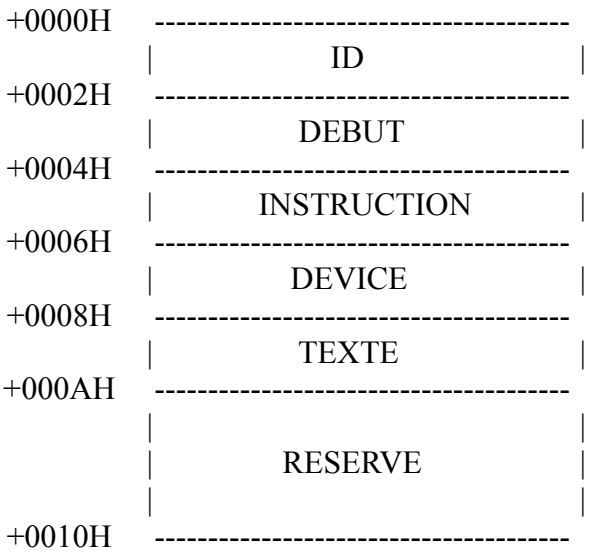
Procédure de recherche d'un programme en cartouche

MSX Basic teste tous les slots (y compris les slots secondaires) de l'adresse 4000H à l'adresse

0BFFFH à la recherche d'un ID valide au début de chaque page, il collationne les informations obtenues et rend le contrôle à chaque page.

Ce test se réalise de gauche à droite (voir le dessin ci-dessus).

Format d'un identificateur :



- ID est un code de 2 octets utilisé pour distinguer les cartouches en ROM des pages pleines.

  'AB' (41H, 42H) est utilisé dans ce but.
- INIT contient l'adresse de la procédure d'initialisation spécifique à la cartouche. (0 quand une telle procédure n'est pas nécessaire).

Les programmes qui nécessitent un travail en coopération avec l'interpréteur Basic devraient retourner sous son contrôle par une instruction RET du Z80 (tous les registres, à l'exception de SP peuvent être détruits). Les autres programmes (comme par exemple les programmes de jeux) n'ont pas besoin de cette procédure.

- INSTRUCTION contient une adresse du traitement étendu d'instruction, à condition qu'une telle adresse soit dans la cartouche (0 dans le cas contraire).

  Quand le Basic rencontre une instruction CALL, il appelle l'adresse ci-dessus dans la zone du système au moyen du nom de l'instruction.

Remarques : (la paire de registres HL sera appelée pointeur de texte.)

- La cartouche doit être installée aux adresses 4000H à 7FFFH.
- La syntaxe pour l'instruction étendue est :

  CALL <instruction_nom> [(<arg> [, <arg>] ..)]

  Un caractère « _ » peut se substituer au mot-clé CALL.
- Le nom de l'instruction est chargé dans la zone du système terminée par 0.

  Le tampon du nom de l'instruction a une longueur fixe de 16 octets. Ainsi ce nom d'instruction ne peut dépasser 15 caractères.
- Si le traitement de cette instruction ne se trouve pas à l'intérieur de la cartouche, il se produit un retour avec le sémaphore de retenue mis à 1.

Le pointeur de texte doit, pour sa part, être retourné inchangé.

- Si le traitement de cette instruction se trouve à l'intérieur de la cartouche, cette dernière devrait :
  - réaliser la fonction
  - mettre à jour le pointeur de texte à la fin de l'instruction (normalement, pointer sur 0 qui indique la fin de la ligne ou sur ':' qui indique la fin de l'instruction)
  - retourner avec le sémaphore de retenue mis à 0. Les registres, à l'exception de SP peuvent être détruits.

  A l'entrée du traitement de l'instruction étendue, le pointeur de texte est activé sur le premier caractère, différent d'un espace, qui apparaît après le nom de l'instruction.

- DEVICE contient l'adresse du traitement de l'instruction étendue si la même se trouve dans la cartouche (0 dans le cas contraire).

  Le Basic appelle cette adresse avec le nom du dispositif dans la zone système.

  Remarques :
  - La cartouche doit être installée aux adresses 4000H à 7FFFH
  - Le nom de l'instruction est chargé dans la zone du système terminé par 0.

    Le tampon du nom de l'instruction a une longueur fixe de 16 octets. Ainsi, ce nom d'instruction ne peut dépasser 15 caractères.
  - une cartouche (16 Ko) peut avoir plus de 4 dispositifs logiques.
  - Quand le Basic rencontre un dispositif qu'il ne reconnaît pas de lui-même, il appelle l'entrée DEVICE avec 0FFH dans A.

    Si le traitement de ce dispositif n'est pas à l'intérieur de la cartouche, le sémaphore devrait être mis à 1 et retourné. Dans le cas contraire, le dispositif ID (de 0 à 3) devrait être retourné dans A et le sémaphore mis à 0.

    Tous les registres peuvent être détruits.
  - Les véritables opérations d'entrée/sortie commencent quand l'entrée DEVICE se présente avec l'une des valeurs suivantes dans A :

0	Ouvert
2	Fermé
4	Entrée/Sortie Random
6	Entrée séquentielle
8	Entrée séquentielle
10	Fonction LOC
12	Fonction LOF
14	Fonction EOF
16	Fonction FPOS
18	Copie d'un caractère

    Le dispositif ID est alors passé dans le DEVICE variable du système.

- TEXTE contient l'adresse de début du texte Basic renfermé dans la cartouche (0 dans le cas contraire).

Le Basic considère ce dernier comme l'adresse de début du texte Basic, pointe vers lui et commence l'exécution du programme.

Remarques :

- Quand il y a plus d'un slot configuré de la sorte, seul celui de l'extrême-gauche (voir dessin ci-dessus) peut être activé puis exécuté.
- La cartouche doit être installée aux adresses 8000H à 0BFFFH. En conséquence, la longueur maximale du texte basic ne peut excéder 16 Ko.
- Si par hasard un bloc RAM occupe les adresses 8000H à 0BFFFH, il ne faut jamais l'utiliser.
- L'adresse pointée par l'entrée TEXTE doit contenir 0.
- Il vaut mieux transférer d'avance vers les pointeurs les numéros des lignes dont les instructions font déjà référence à des numéros de ligne comme GOTO, GOSUB, ... En effet, quand ils seront exécutés, ils ne pourront plus jamais être transférés vers les pointeurs. De toute manière, ils PEUVENT être des numéros de ligne, mais leur exécution serait ralentie.

INIT, INSTRUCTION, DEVICE et TEXTE sont placés dans un ordre croissant, à partir de l'octet bas.

CHAPITRE 7

CONSEILS, BASIC, ET TRUCS

1 Les conseils de programmation de Microsoft

Dans ce chapitre, nous avons regroupé les différents conseils que la société Microsoft donne aux développeurs MSX afin de réaliser des programmes compatibles avec tous les systèmes.

1.1 Comment connaître la version MSX

Il existe différents moyens pour reconnaître la version du MSX sur lequel tourne un programme. En voici quelques-uns :

L'adresse 2DH de la ROM principale contient cette information :

Contenu de 2DH	Version du MSX
0	MSX1
1	MSX2
Autre	Non défini

Pour les programmes tournant avec la page 0 de la ROM principale commutée dans une autre ROM ou RAM (par exemple les programmes sous MSX-DOS), utilisez la fonction de lecture inter-slot à l'adresse 2DH avec l'adresse de ce slot chargée en FCC1H en RAM.

Remarque : Dans un MSX2, la ROM principale n'est pas toujours dans le slot 0 ou dans le slot 0-0. (0-0 désigne le slot secondaire 0 du slot primaire 0).

L'adresse FCC1H contient l'adresse du slot de la ROM principale et l'adresse FAF8H celle du slot de la ROM complémentaire ou Sub-ROM (celle qui contient les spécificités MSX2). Ce fait est dû à l'adaptation du MSX2 au MSX1.

L'adresse FAF8H de la RAM contenant l'adresse de la ROM complémentaire du MSX2 et celle-ci n'existant pas en MSX1, il est possible de savoir si l'on se trouve devant un MSX1 ou pas, suivant le contenu de cette adresse RAM :

Contenu de FAF8H	Version du MSX
0	MSX1
Autre	MSX2

1.2 Adresses d'entrées/sorties du VDP

Dans un MSX2, le VDP n'est pas toujours situé aux adresses 98H-9BH. C'est pourquoi les programmes qui accèdent directement au VDP doivent se référer aux adresses 6 et 7 de la ROM principale pour connaître les adresses du VDP et ceci toujours à cause de l'adaptation du MSX2 au MSX1.

Ports du VDP	Adresses Entrées/sorties
Lecture dans la VRAM	(0006) contenu de l'adresse 6
Ecriture dans la VRAM	(0007)
Etat de la lecture	(0006) + 1
Commande de l'écriture	(0007) + 1
Palette d'écriture	(0007) + 2
Accès indirect au registre	(0007) + 3

1.3 Adresse du slot de la ROM Basic

Dans un MSX2, la ROM principale n'est pas toujours située dans le slot 0 ou dans le slot 0-0. Il existe une autre ROM, appelée Sub-ROM (= ROM complémentaire), contenant les informations complémentaires du MSX2 et placée dans la page 0 d'un des slots. Pour connaître l'adresse de ces slots, il faut consulter les adresses FCC1H et FAF8H de la RAM.

Pour accéder aux entrées du Bios dans ces ROMs, il faut utiliser l'appel inter-slot :

Adresse	Etiquette	Contenu
FCC1H	EXPTBL	Adresse slot ROM principale
FAF8H	EXBRSA	Adresse slot ROM complémentaire

1.4 Slot étendu

Dans une MSX2, il est tout à fait possible d'étendre l'un ou l'autre slot afin d'obtenir plus de place. Ceci lui permettra de contenir plus de programmes en ROM comme par exemple : une ROM complémentaire, une ROM disque, une ROM RS232 ou encore un programme spécial en ROM. Il ne devrait donc jamais y avoir de programme pour un MSX2 qui ne fonctionne pas avec une machine possédant un ou des slots étendus.

Evitez d'accéder à l'adresse FFFFH comme à une adresse RAM normale. Cette adresse sert en effet de registre de sélection du slot étendu. Divers programmes placent malencontreusement le pointeur stack à cette adresse au moyen de l'instruction LD SP, 0000 et ... le programme ne tourne pas sur une machine possédant des slots étendus !

1.5 Appel du Bios

Le Bios doit être appelé par l'intermédiaire des entrées de la table des jumps. Un programme qui appellerait directement l'adresse d'une routine ne serait jamais assuré d'une compatibilité totale.

1.6 Etat initial de la RAM

Les contenus initiaux de la RAM principale et de la VRAM sont indéterminés. Divers logiciels supposent que le contenu de la RAM équivaut à 0 (nous ne savons pas pourquoi) et, bien sûr, ne tournent pas à tous les coups !

1.7 Retour au Basic

Le retour à l'interpréteur Basic, depuis une autre application, se réalise de la manière suivante :

L'espace de travail principal du Basic doit rester inchangé (si vous ignorez ce que sont les espaces de travail, ne changez rien à la zone Basic !)

Le contenu de tous les stacks (piles) et de tous les registres est ignoré.

- Activez le slot de la ROM principale. L'adresse de ce derier est situé en RAM à FCC1H (revoyez au point 3.0 l'adresse du slot de la ROM-Basic).
- Sautez à l'adresse 409BH dans la ROM principale.

Le message « Ok » ou votre propre sigle (MSX2) sera alors affiché.

1.8 Démarrage automatique de logiciels

Pour de simples logiciels comme les jeux par exemple, l'adresse de départ du logiciel doit se trouver à l'emplacement INIT dans la zone ROM ID. De la sorte, aucun autre programme-système comme un disque ou la RS232C ne peut être initialisé.

Pour des applications qui nécessitent, par après, l'initialisation de tout autre programme système, il faut placer l'instruction d'appel inter-slot de l'adresse de départ en FEDAH. Ce vecteur crochet (hook) sera automatiquement appelé après que tous les logiciels su système auront été initialisés. Cette méthode est également utilisable avec un système sans disque.

1.9 Utilisation des vecteurs crochets

Les vecteurs-crochets constituent un des moyens par lequel le Basic MSX peut être étendu. Quelques procédures (comme 'lecture de la console', 'écriture dans la console') possèdent une instruction CALL du Z80 qui est dirigée vers la zone RAM commune. Ces zones sont composées de 5 octets pour l'espace de travail par vecteur-crochet. Elles sont initialisées au moyen de cinq instructions RET du Z80 au démarrage à froid.

Les expansions ne peuvent être réalisées en redirigeant cette entrée ailleurs.

Exemple :

```
                .
                .
                CALL HOOKxx             en ROM
                .
                .
HOOKxx:         RET                             HOOKxx:     RST     6
                RET                     en RAM              DB      <adresse-slot>
                RET                                         DW      <adr-mémoire>
                RET                                         RET
                RET
                RET
```

RST 6 exécute un appel inter-slot vers un autre slot. A ce sujet, voir BIOENT.MAC pour plus de détails au sujet des appels inter-slots.

Pour connecter le vecteur-crochet à la routine désirés, celle-ci dont savoir où elle est, c'est-à-dire dans quel slot elle est. Ceci est très important parce qu'il n'y a aucun indice signalant le slot où la routine est placée.

< ATTENTION >

Une cartouche, dont le programme est écrit en langage machine doit être capable de tourner dans n'importe quel slot, en ce compris les slots secondaires. Il n'existe en effet aucun indice indiquant dans quel slot tourne la cartouche.

1.10 Allocation d'espace de travail

Si le programme tourne seul, c'est-à-dire s'il n'a pas besoin d'un autre programme tournant dans une autre cartouche, toute la RAM en-dessous de la zone de travail du Bios (sous 0F380H) est libre.

Autrement, si le programme tourne avec l'interpréteur Basic et un ou d'autres programmes en cartouche(s), l'utilisation de la RAM est restreinte.

Il existe 3 moyens de partager la RAM à l'usage exclusif de chaque cartouche :

- Mettre une RAM sur la cartouche (meilleur moyen et le plus facile)
- Utiliser le SLTWRK si la zone de travail s'étend sur moins de 3 octets
- Si la zone de travail occupe plus de 2 octets, réaliser le SLTWRK dans la variable système BOTTOM (0FC48H), ensuite l'ajuster suivant l'espace de mémoire nécessaire. BOTTOM est activé par le code d'initialisation au point supérieur de la RAM.

2 Compléments au Basic

Ces quelques lignes décrivent des mécanismes internes du Basic MSX. La bonne compréhension de ces mécanismes peut aider les programmeurs à écrire des routines d'expansion pour les MSX.

2.1 Exploration

En Basic Microsoft, le registre HL est utilisé comme pointeur vers le texte-programme. On l'appelle pointeur de texte. Quand une instruction est exécutée, seul le pointeur de texte doit être convenablement remis à jour. Cela signifie que les autres registres, sauf SP, peuvent être détruits.

CHGTR 4666H

Consulte le pointeur et va chercher le caractère pointé par ce dernier. Les espaces sont ignorés. Le sémaphore de zéro est mis à 0H en fin de ligne ou si un signe « : » est rencontré. Le sémaphore de retenue est mis à 1 si un chiffre (0 à 9) est rencontré.

Juste avant que le Basic appelle le traitement d'instruction, un appel est fait vers CHRGTR de sorte que le sémaphore soit mis à 1 pour refléter le caractère pointé par le pointeur de texte. Cela signifie donc que rien ne suit le nom de l'instruction si un sémaphore de zéro est mis à 1 lorsque le traitement d'instruction est requis.

2.2 Evaluation

FRMEVL 4C64H

L'évaluation commence au caractère pointé par le pointeur de texte. La formule peut être soit numérique, soit alphanumérique.

Dans le cas d'une formule numérique, toutes les combinaisons de nombres entiers, simple précision

et double précision sont permises. Le résultat est forcé au type de composant le plus complexe. C'est pourquoi, si la formule est constituée de nombres entiers, de nombres simple précision et double précision, le résultat sera fourni en double précision.

Dans le cas d'une formule alphanumérique, la formule ne peut être composée qu'avec des éléments de chaînes de caractères sinon une erreur de discordance de type ('Type Mismatch') en résulterait.

En retour de FRMEVL, le pointeur de texte est activé au point du caractère suivant la formule. Le résultat est retourné à VALTYP (VALeur TYPe, 0F663H) et à DAC (Decimal ACcumulator, 0F7F6H) comme suit :

Nombre entier :

VALTYP contient 2

La valeur réelle se trouve dans DAC + 2 et DAC + 3, l'OMS venant en premier lieu.

Simple précision :

VALTYP contient 4

La valeur réelle se trouve dans DAC jusque DAC + 3

Double précision :

VALTYP contient 8

La valeur réelle se trouve dans DAC jusque DAC + 7

Chaîne de caractères :

VALTYP contient 3

DAC + 2 et DAC + 3 contiennent l'adresse du descripteur de chaîne (un descripteur de chaîne consiste en 3 octets dont le premier contient la longueur de la chaîne, le deuxième et le troisième l'adresse de cette chaîne, l'OMS vient en premier lieu).

FRMONT 542FH

Appelle FRMEVL et force le résultat dans un entier signé de 16 bits. Ce dernier est renvoyé via le registre DE.

Une erreur de dépassement (overflow) se produira si la valeur ne peut être représentée par un entier.

GETBYT 521CH

Appelle FRMEVL et force le résultat dans un entier non-signé de 8 bits. Ce sernier est renvoyé via les registres E et A, tous les deux contenant la même valeur.

Avant le renvoi GETBYT réalise un appel vers CHRGTR de sorte que le sémaphore soit mis à 1 pour refléter le caractère devant être pointé par le pointeur de texte.

Une erreur d'appel illégal de fonction (Illegal function call) se produira si la valeur ne se situe pas entre 0 et 255.

FRESTR 64D0H

Lit l'adresse d'un descripteur de chaîne. Le résultat est renvoyé via le registre HL.

FRESTR teste la valeur pour voir si elle est bien du type 'chaîne de caractères'. Si ce n'est pas le cas, FRESTR renvoie une erreur de discordance de type (Type mismatch).

2.3 Renvoi de valeurs

Le renvoi des valeurs peut être réalisé en spécifiant un nom de variable comme argument dans la table des arguments de l'instruction CALL.

Exemple :

Supposons qu'il y ait un 'chip' de timer muni d'une batterie de sécurité et que nous désirons connaître l'heure courante :

CALL RDTIME (A$) ou _RDTIME(A$)

lira l'heure courante et la placera, en tant que chaîne de caractères, dans A$.

PTRGET 5EA4H

Lit le nom de variable débutant à l'endroit où se trouve le pointeur de texte et renvoie ce pointeur à la valeur de champ définie pour cette variable via le registre DE.

Valeurs de VALTYP :

2 → Variable nombre entier
3 → Variable chaîne de caractères
4 → Variable nombre en simple précision
8 → Variable nombre en double précision

La variable peut, être soit une variable ordinaire, soit une variable tableau, définie ou non.

Si cette variable est nouvelle, c'est-à-dire si elle n'a pas encore été définie, une nouvelle zone est créée pour elle.

Si le premier caractère pointé par le pointeur de texte n'est pas une variable-nom, il en résultera une erreur de syntaxe (Syntax error).

Cette routine se termine lorsqu'un caractère ne pouvant être considéré comme une variable ou comme une partie de variable est rencontré. C'est pourquoi il peut s'avérer nécessaire de tester si un champ de variable (Field) se termine de manière appropriée.

3 Trucs et astuces

3.1 Trucs avec le VDP 9938

Scrolling

Voici un petit programme sans prétention qui permet de réaliser un SCROLLING (défilement) vertical. Ce programme utilise le registre 23 du VDP (instruction VDP(24)).

```
10 SCREEN 8 : OPEN « GRP : » AS#1
20 COLOR 10, 0, 0 : CLS
30 CIRCLE (100, 100), 25, 156
40 PAINT (100, 100), 38, 156
50 LINE (50, 50) - (200, 150), 250, B
60 PRINT#1, « ENFONCEZ UNE TOUCHE »
70 IF INKEY$ = « » THEN 70
80 FOR I = 0 TO 255
90 VDP(24) = I
95 NEXT I
99 GOTO 80
```

Mode texte multicolore

Ce petit programme permet d'afficher une marge de couleur rouge autour d'un texte en mode texte 80 caractères en deux couleurs.

```
10 SCREEN 0 : WIDTH 80 : COLOR 10, 0, 0
20 FOR I = &H800 TO &H8EF
30 VPOKE I, 0
40 NEXT I
50 FOR I = &H800 TO &H809
60VPOKE I, 255
70 NEXT I
80 FOR I = &H8E6 TO &H8EF
90 VPOKE I, 255
100 NEXT I
110 FOR I = &H80A TO &H8DC STEP 10
120 VPOKE I, 192
130 VPOKE I + 9, 3
140 NEXT I
150 C = PEEK (6)
160 C = C + 1
170 OUT C, 166
180 OUT C, 140
190 OUT C, 240
200 OUT C, 141
```

3.2 Trucs divers

Ces trucs très simples ont pour but de démontrer tout le parti que l'on peut tirer d'une bonne connaissance du matériel et des routines et variables internes (chapitre 5).

Protection d'un programme anti-list :

Ce truc a pour but de vous démontrer le mécanisme de déroutement des vecteurs crochets. Le vecteur du LIST est situé en FF8AH. Il suffit de remplacer le C9H qui s'y trouve par un saut au OK du Basic (491DH) pour supprimer la possibilité d'utiliser l'instruction LIST.

POKE &HFF8A, &H1D : POKE &HFF8B, &H41 : POKE &HFF8C, &HC3

Verrouillage du BREAK :

Il suffit de laisser croire au système que le Basic est en ROM

POKE &HFBB1, 1

Utilisation des adresses internes pour afficher la date du jour :

PRINT USING « ##-##-20## »; PEEK(&HF248); PEEK(&HF249); PEEK(&HF24A)

Allumage du témoin CAPS et passage en majuscule :

Par l'intermédiaire du port du AB et de la variable interne FCAB

OUT &HAB, 12 : POKE &HFCAB, 255

Reset général :

Truc simple, il suffit de relancer la ROM en 0

DEFUSR=0 : L=USR(0)

Réinitialisation du contenu par défaut des touches de fonction :

Par l'intermédiaire du vecteur Bios documenté dans le chapitre 4

DEFUSR = &H3E : L = USR(0)

4 Le MUSIC MODULE Philips

Pour terminer ce chapitre et par la même occasion ce livre, nous vous livrons les premiers résultats de nos recherches sur le MUSIC MODULE Philips.

Ces informations sont destinées plus aux bidouilleurs de génie qu'aux musiciens.

Le processeur qui équipe le MUSIC MODULE est le Y8950 de YAMAHA programmable en

synthèse FM.

L'interface MIDI est gérée par un circuit ACIA 6850.

La mémoire interne de stockage PCM (Pulse Code Modulation) est de 256 Kbits (32 Ko).

La ROM interne est de 32 Ko.

Le convertisseur digital/analogique est un circuit Yamaha Y3014.

Le processeur central (Y8950) du MUSIC MODULE est interfacé par l'intermédiaire des ports C0H et C1H.

Le port C0H permet de lire le registre d'état et de spécifier en écriture le numéro de registre à atteindre.

Le port C1H permet de transférer les données ou de lire le contenu de certains registres.

Seuls les registres R5, R15, R25 et R26 peuvent être lus.

Le convertisseur 8 bits est situé sur le port 0AH.

Pour l'utiliser, il faut préalablement envoyer 15 (0FH) sur le registre R24 du processeur central.

LE registre R25 permet alors de valider ou d'invalider le convertisseur 8 bits.

L'interface MIDI est accessible par l'intermédiaire des ports 0 (registre de commande et d'état) et 1 (entrée/sortie des données).

Les principaux registres du Y8950 (* : peut être lu)

R0	Non utilisé
R1	Test
R2	Timer 1
R3	Timer 2
R4	Registre général 1
R5*	Entrée clavier
R6	Sortie clavier
R7	Registre général 2
R8	Registre général 3
R9	Partie basse de départ ADD
R10	Partie haute de départ ADD
R11	Partie basse de fin ADD
R12	Partie haute de fin ADD
R13	Partie basse de PRESCALE
R14	Partie haute de PRESCALE
R15*	Donnée ADPCM
R16	Partie haute de DELTA N
R17	Partie basse de DELTA N
R18	CONTROLE EG
R19	Non utilisé
R20	Non utilisé
R21	Données DAC haute
R22	Données DAC basse (B7-6)
R23	SHIFT (B0-B2)
R24	CONTROLE I/O
R25*	Données I/O
R26*	Données PCM
R27-R31	Non utilisé
R32-R53	AM/VIB/EG/KSR/MULTI
R64-R85	KSL/TL
R96-R117	ATTACK-DECAY
R128-R149	SUSTAIN-RELEASE
R160-R168	F NUMBER (haut)
R176-R184	F NUMBER-KON-BLOCK
R189	Registre général 4
R192-R200	FB-C (B0-B3)

ANNEXES

1 Quelques localisations importantes du MSX2

Adresses	Contenus
0006H	Adresse d'entrée/sortie du port de lecture du VDP
0007H	Adresse d'entrée/sortie du port d'écriture du VDP
002DH	Numéro de version MSX de l'appareil utilisé
409BH	Point d'entrée du démarrage à chaud (WARMBOOT) de l'interpréteur Basic
F323H	Point d'accès à la routine du traitement des erreurs disque
FAF8H	Adresse du slot de la ROM complémentaire
FB21H	Informations sur les drivers-disques
FC4AH	Adresse de début de la zone de travail du système
FCC1H	Adresse du slot de la ROM principale
FFA7H	Si le contenu de cette adresse n'est pas 'C9', c'est qu'un disque est connecté
FEDAH	Vecteur-crochet (hook) pour le démarrage automatique
FFFFH	Registre sélection du slot secondaire

2 Séquence ESCAPE acceptées par le MSX

MSX accepte une série de séquences précédées du code ESCAPE (27) qui font partie intégrante du terminale compatible du VT52.

Ces séquences peuvent être envoyées par :

- l'instruction PRINT du Basic
- la routine CHPUT du Bios
- l'appel direct dans le Bios de CONOUT du MSX-DOS
- l'appel de la fonction CONSOL OUTPUT du MSX-DOS

FONCTIONS DU CURSEUR

ESC A	Curseur en haut
ESC B	Curseur en bas
ESC C	Curseur à droite
ESC D	Curseur à gauche
ESC H	Curseur au coin supérieur gauche
ESC Y <ligne + 20H><col. + 20H>	Localisation du curseur

EFFACER ET EDITER

ESC j	Nettoyer l'écran

ESC E	Nettoyer l'écran
ESC K	Effacer jusqu'à la fin de la ligne
ESC J	Effacer jusqu'à la fin de l'écran
ESC l	Effacer une ligne entière
ESC L	Insérer une ligne
ESC M	Effacer une ligne

CONFIGURATION

ESC x4	Affiche un curseur-bloc
ESC x5	Eteint le curseur
ESC y4	Affiche un curseur-trait
ESC y5	Allume le curseur

3 Utilisation de la zone buffer RS232

FAF5 RSIQ	OFFSET	
FAF5 DPAGE	0	Page d'affichage
FAF6 ACPAGE	1	Page active
FAF7 AVCSAV	2	Sauve le port de contrôle AV
FAF8 XBRSA	3	Adresse du slot de la ROM du Basic étendu
FAF9 CHRCNT	4	Compteur de caractères pour Roma-Kana (2 octets)
FAFA ROMA	5	Sauve le caractère
FAFB	6	pour Roma-Kana (2 octets)
FAFC MODE	7	Switch de conversion du mode Roma-Kana avec la VRAM
FAFD	8	Inutilisé - Réservé
FAFE XSAVE	9	Souris - crayon optique
FAFF	10	Sauve X (2 octets)
FB00 YSAVE	11	Souris - crayon optique
FB01	12	Sauve Y (2 octets)
FB02 LOGOPR	13	Code d'opération logique (les 4 bits les moins signi.)
FB03 TOCNT	14	Octet de donnée
FB04 RSFCB	15	Adresse basse de la RS232C
FB05	16	Adresse haute de la RS232C
FB06 RSIQLN	17	Octet de donnée

Zone des données RS 232C

MEXBIH	18	'RST 30H' (0F7H)
	19	Octet de donnée
	20	(bas)
	21	(haut)
	22	'RET' (0C9H)
OLDSTT	23	'RST 30H' (0F7H)
	24	Octet de donnée
	25	(bas)
	26	(haut)
	27	'RET' (0C9H)
OLDINT	28	'RST 30H' (0F7H)
	29	Octet de donnée
	30	(bas)
	31	(haut)
	32	'RET' (0C9H)
DEVNUM	33	Octet offset (décalage)
DATCNT	34	Octet de donnée
	35	Octet pointeur
	36	Octet pointeur
ERRORS	37	Octet de donnée
FLAGS	38	Octet booléen
ESTBLS	39	Octet booléen
COMMSK	40	Octet de donnée
LSTCOM	41	Octet de donnée
LSTMOD	42	Octet de donnée
	43	Utilisé par le système
	63	

FB35H

LE LIVRE DU MSX2

Daniel Martin

avec la participation de M. Devos et G. Gavage

retapé par Granced
(original aimablement fourni par MSXosaure)

pour
la communauté MSX française

LE LIVRE DU MSX 2

Daniel Martin

Avec la participation de M. Devos et G. Gavage

Ce livre réunit toutes les connaissances systèmes acquises sur le MSX 2. Vous y retrouverez notamment, la description complète du générateur sonore, du circuit de visualisation et de digitalisation et de l'horloge temps réel.

Toutes les routines et les variables internes du MSX 2 sont décrites et commentées.

Un chapitre complet est réservé à la description de la manipulation des SLOTS mémoire.

Vous y trouverez également tous les conseils de MICROSOFT réservés jusqu'à présent aux sociétés de développement.

En fin d'ouvrage vous découvrirez enfin quelques trucs et astuces ainsi qu'une description interne du MUSIC MODULE PHILIPS.

BCM s.c.

24, route de la Sapinière - 4960 Banneux Belgique

ISBN 2-87111-011-5

110 FF