

Pratique du

par GDX

Version 2016.03 (pré-version)

Basé sur le livre « Pratique du MSX2 »

de Eric Von Ascheberg

(original aimablement fourni par Metalion puis
retapé par Granced pour
la communauté MSX française)

Informations complémentaires tirées du

MSX Datapack vol. 1 à 3 de ASCII

et de

divers sites sur Internet.

Index

1 Le standard MSX	4
1.1 Introduction.....	4
1.2 Utiliser ce livre.....	5
1.3 Configurations minimales.....	6
1.4 Conseils au développeur de logiciels.....	8
1.5 Et maintenant.....	8
2 Les Slots et le Memory Mapper	10
2.1 Comment dépasser la limite des 64 Ko.....	11
2.2 Qu'est-ce qu'un Slot ?.....	11
2.3 Utiliser les Slots.....	14
2.4 Le Memory Mapper.....	17
3 Le Bios (Basic Input/Output System)	19
3.1 Introduction au Bios.....	19
3.2 Table des sauts du Bios en Main-ROM.....	19
3.3 Table des sauts du Bios de la Sub-ROM.....	68
3.4 Table des sauts du Bios de la Disk-ROM.....	101
3.5 Le Bios des extensions MSX.....	106
4 Les variables système et zones de travail	121
4.1 Introduction aux variables système.....	121
4.2 La liste des variables et des zones de travail système.....	121
4.3 Quelques exemples d'utilisation des variables système.....	134
5 Les Hooks	136
5.1 La liste des Hooks.....	136
6 Le processeur vidéo (VDP)	151
6.1 Avertissement.....	151
6.2 Introduction au processeur vidéo.....	151
6.3 Fonctionnement du VDP.....	151
6.4 Comment accéder au VDP.....	152
Ecrire dans un registre de contrôle (0 ~ 23, 25 et 32 ~ 46).....	153
Ecrire dans un registre de la palette des couleurs.....	155
Lire un registre de statut du VDP.....	156
6.5 Lecture et écriture dans la mémoire vidéo.....	157
6.6 Les registres de contrôle du processeur vidéo.....	158
6.7 Les registres de statut du VDP.....	174
6.8 Les commandes internes du V9938 et V9958.....	176
HMMC (High speed Move to Memory from CPU).....	179
YMMM (Y Move to Memory from Memory).....	181
HMMM (High Speed Move to Memory from Memory).....	183
HMMV (High speed Move to Memory from VDP).....	185
LMMC (Logical Move to Memory from CPU).....	187
LMCM (Logical Move to CPU from Memory).....	189
LMMM (Logical Move to Memory from Memory).....	191
LMMV (Logical Move to Memory from VDP).....	193
LINE (draw a LINE).....	195
SRCH (SeaRCH for color).....	197
PSET (Point SET).....	199
INSTRUCTION POINT (is POINT set ?).....	201
6.9 Les différents modes d'affichage (SCREEN 0 à SCREEN 12).....	202
6.10 Les Sprites et leur fonctionnement.....	219
6.11 La souris et le crayon optique.....	229
7 Des applications types	230
7.1 Revenir à l'interpréteur Basic.....	230
7.2 Quel type de MSX ?.....	230
7.3 Le « PRINT » en assembleur.....	231

7.4	Système a disquettes ou pas ?.....	231
7.5	Traiter les erreurs liées aux disquettes.....	232
7.6	Faire de la musique en assembleur.....	232
7.7	Passage de paramètres du Basic au langage machine.....	233
7.8	Ajouter une instruction au Basic avec CMD ou IPL.....	236
7.9	Les codes de contrôle CTRL.....	240
7.10	Les codes escape [ESC].....	240
7.11	Utiliser le second jeu de caractères.....	242
7.12	Trouver de la RAM sur les plages 0 et 1.....	246
7.13	Détourner le Reset.....	248
7.14	Ajouter des mots clés au Basic.....	249
7.15	Manipuler la souris.....	252
7.16	Développer un programme en cartouche.....	253
8	Le MSX-DOS	255
8.1	Zone fixe de travail et des variables du MSX-DOS1 et du Disk-BASIC.....	256
8.2	Zone fixe de travail et des variables du MSX-DOS2 et du Disk-BASIC 2.....	262
	Annexes	269
A	- Les BIOS.....	269
B	- Les Variables Système.....	273
C	- Les Hooks.....	277
D	- Les registres du VDP.....	278
E	- Les cartes de la mémoire vidéo.....	282
F	- Jeux de caractères MSX.....	285
G	- Exemples de matrices de clavier.....	289
H	- Codes d'erreur du Basic et du disque Basic.....	292

1 Le standard MSX

1.1 Introduction

Le standard MSX n'a pas connu le succès escompté en Europe. Si l'on pouvait faire des reproches justifiés à la première version (prix trop élevé des premiers modèles, mémoire vive un peu juste, etc), la version 2 était un ordinateur assez puissant et convivial. Il a cependant souffert de concurrence avec les ordinateurs 16 bit qui apparurent presque aussi tôt.

En effet, le MSX 2 représentait, en toute simplicité, l'ordinateur 8 bit familial le plus puissant jamais construit ! Bien des machines professionnelles de l'époque ne possèdent pas les caractéristiques du MSX2. Citons, pour mémoire, les principales du modèle le plus vendu :

- 256 Ko de mémoire vive extensible à 4 Mo par Slot libre
- 128 Ko de mémoire vidéo extensible à 192 Ko
- 64 Ko de mémoire morte comprenant notamment un Basic Microsoft très évolué
- 2 Slot d'extension, possibilité de passer à 8 Slot
- lecteur de disquette 1 Mo (720 Ko formatée) intégré
- gestion de deux lecteurs de disquette simultanément (jusqu'à huit possible)
- compatibilité totale IBM PC au niveau des fichiers (FAT12)
- horloge interne alimentée par pile
- mémoire CMOS comprenant un système de mots de passe
- processeur graphique VLSI
- résolution 256 sur 212 pixels en 256 couleurs simultanément
- souris
- deux systèmes d'exploitation de disquettes (dont un possédant l'interface menus déroulants/icônes)
- générateur sonore sur trois voix, huit octaves
- 700 logiciels disponibles, dont une trentaine spécifique MSX2
- de très nombreux périphériques

Il va de soi que la maîtrise du MSX ne se limite pas à une bonne connaissance du Z80 (micro-processeur qui équipe tous les MSX). Il est nécessaire, comme sur tous les ordinateurs, de pouvoir mettre en œuvre tous les composants du système. De plus, le programmeur qui travaille sur MSX doit veiller à maintenir la compatibilité, y compris avec les futurs modèles. C'est le but de ce livre qui se propose de vous guider à travers l'exploration de la programmation du système MSX.

1.2 Utiliser ce livre

Le livre que vous tenez entre vos mains a été conçu dans une double optique. Il aspire, dans un premier temps, à apprendre à tout programmeur ayant une bonne connaissance du Basic et des notions sérieuses en Z80, à tirer le maximum de son ordinateur, qu'il s'agisse d'un MSX1 ou plus récent. A ce propos, il est précisé à chaque fois que cela est nécessaire si les explications s'adressent à une version particulière de MSX ou à différentes versions. Puis, une fois les notions assimilées, le présent ouvrage a la prétention de servir de manuel de référence complétant ainsi le manuel d'utilisation fourni avec votre machine et ce, même pour les versions de MSX qui ne sont jamais sortis en France.

Comprendre le fonctionnement :

Pour chaque fonction, instruction, astuce, j'ai tenté de donner les explications les plus simples possibles. J'ai évité d'employer l'équivalent français de termes anglais lorsque ces derniers étaient usuels (je parle de « bitmap » plutôt que d'image matricielle, mais j'emploie le mot « octet » et non « byte »). Lorsqu'une traduction paraissait hasardeuse, j'ai toujours ajouté le terme américain entre parenthèses.

De plus, suivant le vieil adage des informaticiens « rien ne vaut la pratique », presque toutes les explications se trouvent accompagnées d'un exemple de programme Basic ou en langage assembleur l'illustrant. Lisez l'explication, si la compréhension n'est pas immédiate, essayez de taper le programme qui l'accompagne. Si vous ne voyez toujours pas, votre cas est sans espoir.

Certains points ne sont pas abordés dans ce livre. Il s'agit tout d'abord du micro-processeur. Si vous avez des problèmes avec cet élément, je ne peux que vous conseiller l'achat du « Programmation du Z80 » de Rodney Zaks chez Sybex, véritable « Bible » du Z80 de plus de six cent pages (en anglais). De même, je ne m'étends pas sur le fonctionnement du processeur sonore PSG (« Programmable Sound Generator »). Tous les manuels livrés avec les différents modèles de MSX donnent les renseignements nécessaires à la programmation de ce composant. Voyez tout de même le paragraphe 6.6, « [Faire de la musique en assembleur](#) », du chapitre 6.

Ce manuel ne comporte aucune information sur le matériel même (« hardware »). Il n'aborde les différents éléments qu'au niveau logiciel (« software »). Il s'adresse donc avant tout au programmeur et non à l'électronicien.

Retrouver un renseignement rapidement :

Lorsque vous maîtrisez l'application qui vous intéresse, évitez tout ce qui est note, remarque, etc. En général, les renseignements indispensables à la mise en œuvre de l'application (adresses mémoires, registres à charger, etc) se trouvent en début de paragraphe, les explications venant après. Il va de soi qu'il peut être utile de consulter les programmes donnés en exemple, ils permettront en effet souvent un gain de temps appréciable.

A la fin de l'ouvrage sont réunies six annexes qui renferment une grande partie des petites choses dont un programmeur a toujours besoin et qui ne se trouvent, bien entendu, jamais là où on les cherche. Essayez d'exploiter ces annexes au maximum.

Dans tous les cas, n'hésitez pas à corriger les erreurs et à compléter les oublis qui ne manqueront pas de se révéler. Si vous avez le temps, [envoyez-moi tous les conseils, remarques et autres critiques](#) que la lecture de cet ouvrage vous aura inspirés. Je vous en remercie d'avance.

1.3 Configurations minimales

Afin de faire un programme qui s'adresse à un maximum d'utilisateurs, il est utile de connaître la configuration minimale pour chaque version du standard MSX.

MSX1 :

- Un microprocesseur central (CPU) Z80A ou compatible, 8 bits, cadencé à 3,579545Mhz.
- 8Ko de RAM.
- 32Ko de ROM (Main-ROM)
- Un processeur vidéo (VDP) TMS9918A/TMS9918 (NTSC) ou TMS9929 (PAL) de Texas Instrument ou compatible avec 16Ko de VRAM dédiée.
- Un processeur sonore AY-3-8910 de General Instrument ou équivalent, un PSG (Programmable Sound Generator) à 3 voix sur 8 octaves avec générateur de bruit.
- Une interface programmable pour les ports E/S (PPI) compatible avec le 8255 d'Intel.
- Un port cartouche ou un Bus d'extension de type Slot. (Les broches +12V, -12V et l'entrée sonore sont optionnelles sur le Bus d'extension).
- Un clavier doté de 10 touches de fonction plus 4 touches de déplacement du curseur.
- Différents connecteurs : sortie vidéo (RCA, RGB ou autres), magnéto cassettes, 1 général (manette de jeu, souris, Paddle Controler, Trackball, etc).

MSX2 :

- Un microprocesseur central (CPU) Z80A ou compatible, 8 bits, cadencé à 3,579545Mhz.
- 64Ko de RAM.
- 48Ko de ROM (Main-ROM et Sub-ROM)
- Un processeur vidéo v9938 de ASCII, Microsoft et Yamaha avec 64Ko de VRAM dédiée.
- Un processeur sonore AY-3-8910 de General Instrument ou équivalent, un PSG (Programmable Sound Generator) à 3 voix sur 8 octaves avec générateur de bruit.
- Une interface programmable pour les ports E/S (PPI) compatible avec le 8255 d'Intel.
- Un port cartouche ou un Bus d'extension de type Slot. (Les broches +12V, -12V et l'entrée sonore sont optionnelles sur le Bus d'extension).
- Un clavier doté de 10 touches de fonction plus 4 touches de déplacement du curseur.
- Différents connecteurs : sortie vidéo (RCA, RGB ou autres), magnéto cassettes, 1 général (manette de jeu, souris, molettes (« Paddle Controler »), Trackball, etc).

MSX2+ :

- Un microprocesseur central (CPU) Z80A ou compatible, 8 bits, cadencé à 3,579545Mhz.
- 64 Ko de RAM (Memory Mapper).
- 96 Ko de ROM (Main-ROM, Sub-ROM, Disk-ROM et Kanji Driver).
- 128 Ko de ROM, police de Kanji (Kanji ROM niveau 1).
- Un processeur vidéo v9958 de ASCII, Microsoft et Yamaha avec 64Ko de VRAM dédiée.
- Un processeur sonore AY-3-8910 de General Instrument ou équivalent, un PSG (Programmable Sound Generator) à 3 voix sur 8 octaves avec générateur de bruit.
- Une interface programmable pour les ports E/S (PPI) compatible avec le 8255 d'Intel.
- Deux ports cartouche de type Slot.
- Un clavier doté de 10 touches de fonction plus 4 touches de déplacement du curseur et d'un pavé numérique .
- Différents connecteurs : sortie vidéo (RCA, RGB ou autres), magnéto cassettes, 2 général (manette de jeu, souris, molettes (« Paddle Controler »), Trackball, etc).

MSX Turbo-R :

- Un microprocesseur (CPU) compatible Z80A de Zilog, 8 bits, cadencé à 3,579545Mhz
- Un CPU R800 de ASCII, 16 bits, cadencé à 7.15909MHz.
- 256 Ko de RAM (Memory Mapper).
- 160 Ko de ROM (Main-ROM, Sub-ROM, Disk-ROM, Kanji Driver, MSX-DOS2 ROM et MSX-Music).
- 256 Ko de ROM, police de Kanji (Kanji ROM niveau 1 et 2).
- Un processeur vidéo v9958 de ASCII et Yamaha avec 128Ko de VRAM dédiée.
- Un processeur sonore AY-3-8910 de General Instrument ou équivalent, un PSG (Programmable Sound Generator) à 3 voix sur 8 octaves avec générateur de bruit.
- Un PCM capable de numériser et restituer des sons jusqu'à 15 Khz.
- Un processeur sonore YM2413 de Yamaha, 9 voix FM ou bien 6 + 5 de rythme.
- Une interface programmable pour les ports E/S (PPI) compatible avec le 8255 d'Intel.
- Deux ports cartouche de type Slot.
- Un clavier doté de 10 touches de fonction plus 4 touches de déplacement du curseur et d'un pavé numérique.
- Différents connecteurs : sortie vidéo (RCA, RGB ou autres), imprimante, lecteur de disquettes, 2 ports généraux (manette de jeu, souris, Trackball, etc).

Note : Le port du magnéto-cassette a été retiré. Le Turbo R n'est plus compatible avec les molettes (« Paddle Controler ») ASCII.

1.4 *Conseils au développeur de logiciels*

- Les programmes que vous écrivez ne doivent en aucun cas appeler directement les routines dans la mémoire morte. Celle-ci varie d'une marque à l'autre et surtout des MSX1 aux MSX2. Il est absolument indispensable de passer par la table des sauts du Bios. Chaque Rom (Main ROM, Sub-ROM et ROM Disk) possède un Bios avec sa table des sauts.
- Pour maintenir la compatibilité entre les différents MSX, il ne faut accéder au matériel sans passer par le Bios. Ce dernier constitue une sorte de tampon entre les programmes et le matériel.
- Une des exceptions à la règle ci-dessus concerne le processeur vidéo. Pour des raisons de vitesse d'exécution, vos programmes peuvent adresser le VDP (« Video Display Processor ») sans passer par le Bios. Les adresses mémoire 00006h et 00007h contiennent l'adresse du port de lecture et du premier port d'écriture du processeur vidéo. Voir le [chapitre 6](#) pour plus de précisions à ce sujet.
- Ne pas utiliser la mémoire vive située au dessus de 0F380h comme de la RAM ordinaire. En effet, cette zone mémoire contient les variables système (voir [chapitre 4](#)) indispensables à la bonne marche de votre ordinateur.
- Il existe des différences entre les MSX commercialisés en France et dans les autres pays - ne serait-ce que le clavier - dont il est parfois important de tenir compte. Les cases mémoire 0002Bh et 0002Ch donnent des renseignements importants à ce sujet. Voir le [chapitre 3.2](#).
- Sur MSX2 uniquement, vous trouverez des renseignements complémentaires dans la mémoire CMOS de l'horloge. Voir la routine REDCLK (001F5h), [chapitre 3.3](#) « Table des sauts du Bios en Sub-ROM ».
- Certains programmeurs placent la pile en haut de mémoire avec l'instruction LD SP, 00000h. Ceci ne fonctionne évidemment pas sur MSX car l'adresse 0FFFFh est utilisée pour accéder aux registres des Slots secondaires (voir le [chapitre 2](#) à propos des Slots).
- Certains programmeurs partent du principe que la mémoire vive principale ou vidéo contient 0 à l'allumage. Il va de soi que ce n'est pas le cas et que la RAM peut contenir à peu près n'importe quoi si elle n'a pas été modifiée lors de l'initialisation du système.
- Si votre programme ne peut fonctionner avec la présence d'un lecteur de disquettes, il suffit de vérifier la présence du lecteur (voir le [chapitre 7](#)) puis, si un lecteur est bien présent, d'afficher à l'écran le message « Faites un RESET en laissant la touche SHIFT enfoncée jusqu'au bip sonore ».
- En effet, lors de l'initialisation du système, si le MSX détecte que la touche SHIFT est enfoncée, il n'installe pas les lecteurs de disquettes.
- Sur le même principe, la touche CTRL assigne un seul lecteur par contrôleur. Si vous avez 2 lecteurs de disquettes avec 2 contrôleurs, lors de toutes les opérations, le premier s'appelle « A » alors que le second prend la dénomination « C ». En enfonçant CTRL à l'initialisation, vous gagnerez de la place mémoire et vos lecteurs se nommeront « A » et « B ».
- Lorsque l'on écrit dans un registre « Write only » du processeur vidéo, il est préférable de sauvegarder la donnée dans la zone des variables systèmes adéquate. Ainsi, la donnée pourra être relue à tout moment, ce qui serait impossible autrement.

1.5 *Et maintenant...*

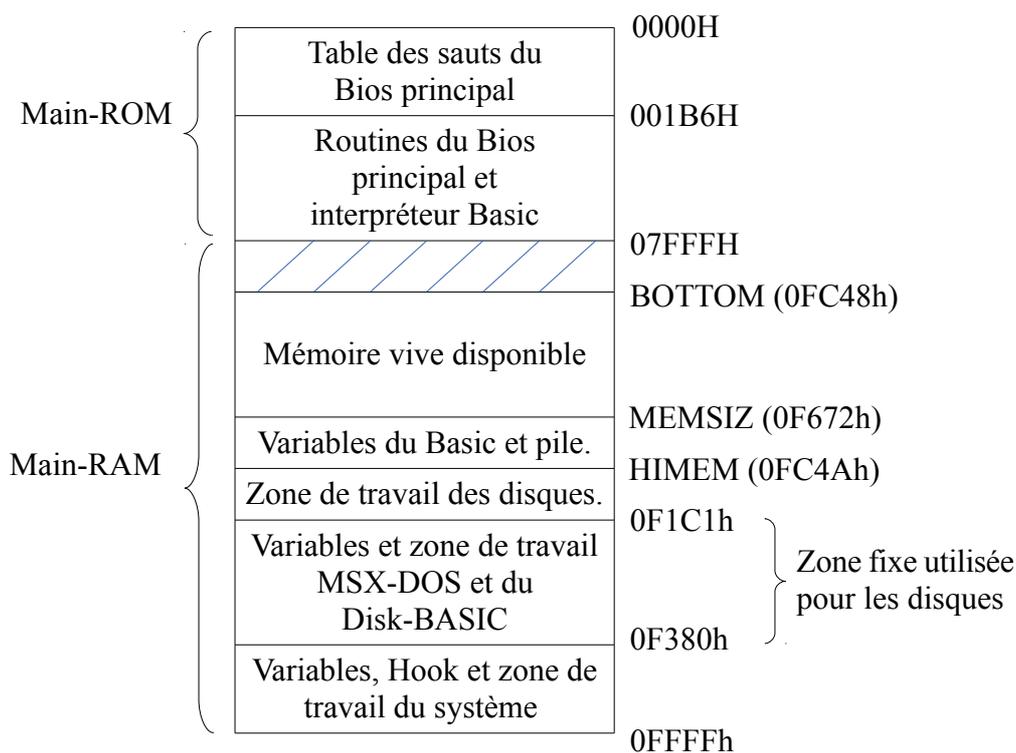
A présent, vous allez vous lancer dans la découverte des Slots, Bios, variables systèmes, et autres processeurs vidéo. Une bonne dose de patience sera, dans la plupart des cas, appréciable et bénéfique. Et maintenant à vous de jouer... et bonne chance !

2 Les Slots et le Memory Mapper

La mémoire vive principale (Main-RAM) est la mémoire sélectionnée au démarrage pour y installer le système. Dans l'environnement du Basic, cette mémoire est disposée de la façon suivante.

Carte de la mémoire principale sous Basic

Cette carte doit être respectée afin que votre programme n'empiète pas sur une zone autre que celle réservée à l'utilisateur (Mémoire vive disponible).



Cette carte montre que le MSX sélectionne seulement la partie haute de la Main-Ram. La mémoire vive disponible pour l'utilisateur commence à l'adresse 0E000h ou 0C000h sur les MSX de 8 ou 16Ko. Elle commence à l'adresse 08000h sur tous les autres MSX. La variable système BOTTOM (0FC48h) indique cette adresse. Pour connaître la fin de mémoire vive disponible, il faudra lire la variable HIMEM (0FC4Ah). L'instruction CLEAR du basic permet de créer une zone « protégée » entre celle de la pile et celle de travail des disques afin d'y mettre nos propres routines en langage machine.

Sur les MSX ayant plus de 32Ko de mémoire vive, il est nécessaire de manipuler les Slots pour accéder à cette mémoire qui est inaccessible au Basic. Ce chapitre explique comment faire.

2.1 Comment dépasser la limite des 64 Ko

Tous les ordinateurs MSX sont équipés d'un Z80 ou compatible comme micro-processeur principal. Ce dernier est un processeur « 8 bits », ceci signifie qu'il manipule les données par paquets de 8. Quant aux adresses mémoires, elles se trouvent codées sur 2 mots de 8 bits (deux octets), soit 16 bits. Les adresses peuvent donc prendre n'importe quelle valeur entre 0000000000000000 et 1111111111111111 en binaire, ou 0 et FFFF en hexadécimal. Ceci équivaut en décimal à une valeur entre 0 et 65535. Sachant qu'un « kilo » informatique ne vaut pas 1000 mais 2 puissance 10 soit, 1024 octets (un des petits secrets qui fait le charme de l'informatique), le Z80 qui peut accéder à 65536 adresses, gère donc 65536/1024, soit 64 kilo octets.

Ce que « voit » le processeur



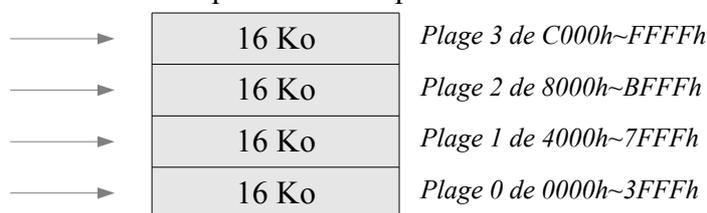
Mémoire centrale

Ainsi, quoiqu'il arrive, le Z80 ne pourra jamais adresser plus de 64 Ko de mémoire. Les concepteurs du MSX ont donc introduit un système, appelé « Slot », qui sépare la mémoire sur quatre plages (Bank en anglais) de 16 Ko afin de en changer le contenu à volonté. Avec ce dispositif, l'utilisateur peut activer la mémoire d'un autre Slot plage par plage.

Toutes ces opérations de manipulations de mémoire restent invisibles au Z80 qui utilise ses 64 Ko comme s'il n'y avait pas d'autre mémoire.

On numérote chaque plage de 16 Ko de 0 à 3.

Ce que « voit » le processeur



Mémoire centrale

Il existe un autre dispositif autorisant la manipulation de pages de mémoire dans les Slots, c'est le « Memory Mapper ». Les « Slot » sont utilisés sur tous les MSX. Quant au « Memory Mapper », il est apparu avec le MSX2. Seuls certains sont compatibles MSX1 (voir plus bas).

2.2 Qu'est-ce qu'un Slot ?

Un Slot est une connexion qui permet de relier une mémoire à un dispositif comprenant en quelque sorte des « aiguillages » à quatre positions au lieu de la relier directement au CPU. Chaque Slot est divisé par quatre afin de pouvoir sélectionner uniquement 16Ko de mémoire sur une plage mémoire comprise entre 0000h et 3FFFh (plage 0), 4000h et 7FFFh (plage 1), 8000h et BFFFh (plage 2), ou C000h et FFFFh (plage 3).

Il peut y avoir deux sortes de Slot, les Slots primaires et les Slots secondaires. Ces derniers sont reliés à un Slot primaire et fonctionnent de façon similaire. Un MSX peut avoir jusqu'à 4 Slot primaires dont chacun peut être relié à 4 Slots secondaires au lieu d'une mémoire. Ce dispositif permet de gérer 16 fois

plus de mémoire.

Voici un exemple d'un MSX ayant ses 4 Slots primaires étendus avec sa Main-ROM dans le Slot secondaire 0-0, une ROM quelconque dans le Slot secondaire 3-0 et ces 32Ko de RAM dans le Slot secondaire 3-2 :

Slot Primaire 0				Slot primaire 1				Slot primaire 2				Slot primaire 3				
SS0	SS1	SS2	SS3													
														RAM		Plage 3
														RAM		Plage 2
ROM												ROM				Plage 1
ROM												ROM				Plage 0

La mémoire centrale (ce que voit le CPU) sera déterminée par le numéro de Slot pour chaque plage.

Plage 3 du Z80	3-2	Mémoire du Slot 3-2.
Plage 2 du Z80	3-2	Mémoire du Slot 3-2.
Plage 1 du Z80	0-0	Mémoire du Slot 0-0.
Plage 0 du Z80	0-0	Mémoire du Slot 0-0.

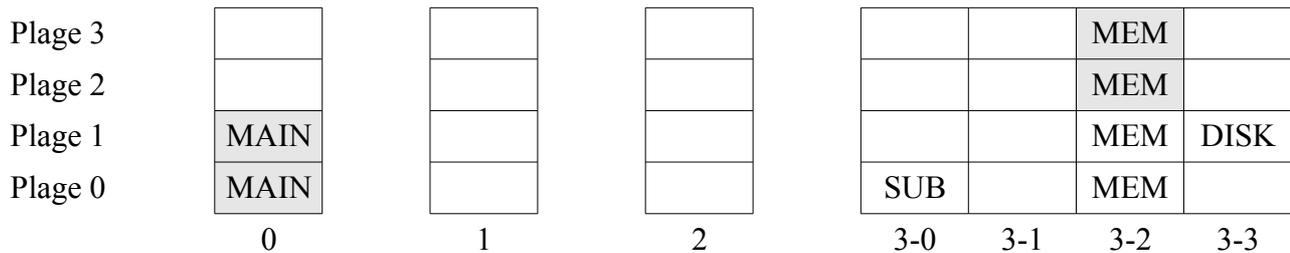
Mémoire centrale

Voici un exemple de la disposition par défaut sur un MSX1, le Philips VG-8020/19 :

Plage 3													
Plage 2													
Plage 1			MAIN										
Plage 0			MAIN										
	0	1	2	3-0	3-1	3-2	3-3						

- Au démarrage sous Basic, les plages 0 et 1 contiennent le Bios et l'interpréteur du Basic qu'il y a dans la mémoire morte principale (Main-ROM). Celle-ci se trouve dans le Slot primaire 0.
- Les Slots 1 et 2 correspondent aux deux ports cartouches du 8020. Le contenu d'un jeu en cartouche pourra être lu en sélectionnant le Slot 1 ou 2.
- Les plages 0 à 3 du Slot 3-2 contiennent chacune 16 Ko de mémoire vive. On comprend pourquoi on ne dispose que de 32 Ko (même un peu moins) sous Basic. Le Z80 « voit » le Bios et l'interpréteur Basic en pages 0 et 1. Il ne reste donc que les pages 2 et 3 pour de la mémoire vive.

Voici un exemple de disposition sur un MSX2, le VG-8235 de Philips :



Quelques remarques :

- Au démarrage sous Basic, les plages 0 et 1 contiennent le Bios et l'interpréteur du Basic qu'il y a dans la mémoire morte principale (Main-ROM) du Slot 0.
- Les Slots 1 et 2 correspondent aux deux ports cartouches.
- La plage 0 du Slot 3-0 renferme la ROM auxiliaire (Sub-ROM) qui contient un Bios pour manipuler les fonctions supplémentaires spécifiques au MSX2 ou plus récent.
- Les plages 0 à 3 du Slot 3-2 contiennent chacune 16 Ko de mémoire vive. (Les autres 64 Ko de la mémoire vive totale qui compte 128 Ko sur le VG-8235 sont accessibles via le Memory Mapper que nous verrons plus loin dans ce chapitre.)
- La Disk-ROM en plage 1 du Slot 3-3 contient les routines du DOS et du Disk-Basic. (Le VG-8235 est équipé d'un lecteur de disquettes intégré.)

La configuration minimum que l'on est certain de trouver sur tout ordinateur se compose de :

MSX1 :

- Une demi page de mémoire vive (située de 0E000h à 0FFFFh) dans n'importe quel Slot.
- Deux pages de mémoire morte (située de 00000h à 07FFFh) contenant le MSX-Basic version 1.0 et le Bios de la ROM principale (« Main-ROM ») dans n'importe quel Slot.
- Un port cartouche ou un Bus d'extension occupant un Slot primaire.

MSX2 :

- Quatre pages consécutives (64 Ko) de mémoire vive dans n'importe quel Slot. (En général dans un même Slot mais pas toujours)
- Trois pages (48 Ko) de mémoire morte divisée en deux parties la « Main-ROM » et la « Sub-ROM », dans n'importe quel Slot .
- Un port cartouche occupant un Slot primaire.

2.3 Utiliser les Slots

Dans toutes les routines du systèmes, le numéro de Slot est codé sur un octet de la manière suivante :

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
EXT	-	-	-	SS1	SS0	PP1	PP0

PP1 et PP0 = Ces deux bits correspondent au numéro de Slot primaire (de 0 à 3).

SS1 et SS0 = Ceux-ci correspondent au numéro de Slot secondaire (de 0 à 3).

EXT = Le bit de poids fort de l'octet est un indicateur de type de Slot. Ce bit s'agit d'un Slot secondaire (EXT à 1), ou alors d'un Slot primaire (EXT à 0).

Les bits 4 à 6 varient suivant le contexte et n'ont aucun rapport avec les Slots.

Les variables systèmes liées aux Slots :

– **EXPTBL** (0FCC1h~0FCC4h)

Ces quatre variables systèmes indiquent chacune si le Slot primaire correspondant est étendu en Slot secondaire (octet = 080h) ou s'il ne l'est pas (octet = 000h).

– **SLTTBL** (0FCC5h~0FCC8h)

Ces quatre variables systèmes retiennent, en complément de EXPTBL, la valeur des 4 registres des Slots secondaires lorsqu'un Slot primaire est étendu.

– **SLTATR** (0FCC9h~0FD08h)

Les 64 octets de SLTATR informent sur le contenu des ROM de chaque Slot. Quatre plages de mémoire de 16 Ko réparties sur 16 Slot secondaires possibles, ce qui fait 4×16 Slot.

Format de ces octets :

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
BT	DE	SE	-	-	-	-	-

Les 3 bits de poids fort sont défini en fonction de l'entête de la ROM logée au Slot correspondant.

SE (« Statement Expander ») = ROM contenant des instructions Basic étendus CALL si à 1.

DE (« Device Expander ») = ROM contenant un programme pour gérer un dispositif si à 1.

BT (« Basic Text ») = ROM contenant un programme Basic si à 1.

Les 5 bits de poids faible ne sont pas utilisés. Ils sont en général à 0 mais peuvent contenir n'importe quoi.

Voici un petit programme Basic qui donne un descriptif de votre ordinateur :

```
10 SCREEN 0:WIDTH80:COLOR 10,0,0:CLS
20 FOR I=&HFCC9 TO &HFD08
30 IF PEEK(I)=0 THEN NEXT:END
40 A=I-&HFCC9
50 PRINT"Page"+STR$(A MOD4)+SPACE$(2)+"Slot"+STR$(INT(A/16))+"-"+STR$(A/4)
MOD 4)+": ";
60 IF (PEEK(I)AND &H80)=128 THEN PRINT TAB(20)+"Extension Basic."
70 IF (PEEK(I)AND &H40)=64 THEN PRINT TAB(20)+"Extension Matériel."
80 IF (PEEK(I)AND &H20)=32 THEN PRINT TAB(20)+"Extension Logiciel."
90 PRINT:NEXT
```

- **SLTWRK** (0FD09h)

Les 128 octets de SLTWRK constituent une zone de travail. Deux octets sont réservés à chacune des 64 pages.

- **MNROM** (0FCC1h)

Sur MSX2, cet octet contient le numéro de Slot qui contient la Main-ROM.

- **EXBRSA** (0FAF8h) (MSX2 ou plus récent.)

Cet octet donne le numéro du Slot qui contient la Sub-ROM.

Les routines du Bios liées aux Slot :

- **RDSL**T (ReaD Slot)

Adresse : 0000Ch (Main-ROM)

Rôle : Sélection d'un Slot et lecture d'une case mémoire.

- **WRSL**T (WRite Slot)

Adresse : 00014h (Main-ROM)

Rôle : Sélection d'un Slot, puis écriture dans une case mémoire.

- **CALSL**T (CALl Slot)

Adresse : 0001Ch (Main-ROM)

Rôle : Appel inter-Slot à une adresse.

- **ENASL**T (ENable Slot)

Adresse : 00024h (Main-ROM)

Rôle : Sélection d'un Slot.

- **CALLF** (CALL Far)
 - Adresse : 00030h (Main-ROM)
 - Rôle : Appel inter-Slot à une adresse.

- **RSLREG** (Read primary Slot REGister)
 - Adresse : 00138h (Main-ROM)
 - Rôle : Lecture du registre des Slots primaires.

- **WSLREG** (Write Slot REGister)
 - Adresse : 0013Bh (Main-ROM)
 - Rôle : Écriture dans le registre des Slots primaires.

Ces routines n'appellent pas de commentaire particulier (voir le Bios) sauf peut-être la routine CALLF « CALL Far » :

Pour accéder à une adresse se trouvant dans un autre Slot, il suffit d'écrire les 3 instructions suivantes en assembleur :

```
RST 30
DB numéro du Slot
DW adresse à appeler
```

Le RET de la routine appelée renverra l'exécution à l'octet immédiatement après l'adresse définie par le DW. Par exemple, en langage machine, la suite d'octets F7h, 8Eh, 0h, C0h, AFh générerait la séquence suivante :

```
0F7h instruction « RST 30 » du Z80
08Eh numéro de Slot secondaire 2-3
000h adresse à appeler 0C000h
0C0h
0AFh instruction « XOR A » du Z80, l' instruction RET de la routine en 0C0000h dans le
Slot 2-3 fera reprendre l'exécution du programme à ce « XOR A ».
```

2.4 Le Memory Mapper

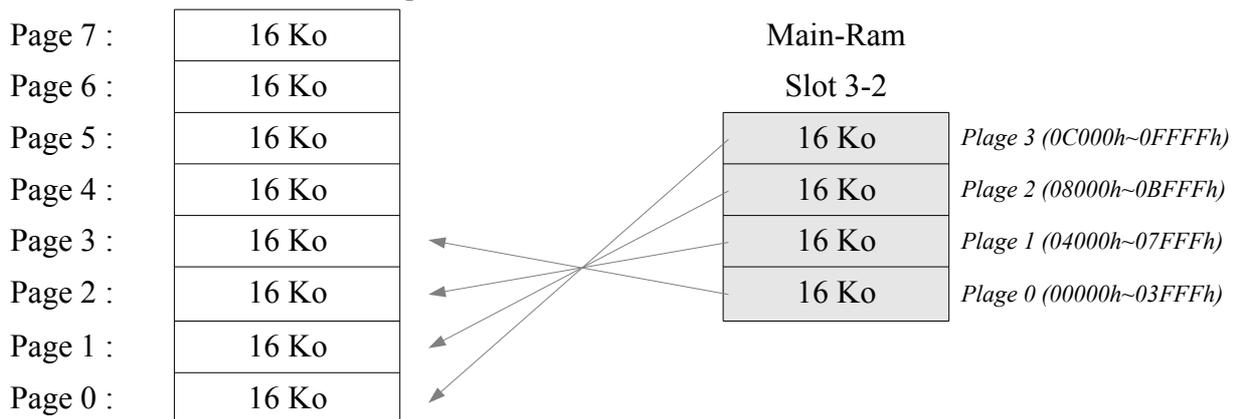
La plupart des MSX2 disponibles en France se trouvent équipés d'un dispositif perfectionné permettant de gérer des quantités assez importantes de mémoire vive (128 ou 256 Ko de RAM sur les modèles disponibles chez nous, mais pouvant aller en théorie jusqu'à 4 Mo par Slot).

La mémoire d'un Memory Mapper est manipulable par page de 16Ko. Chaque page d'un Memory Mapper pouvant être placée sur n'importe quelle des quatre plages mémoire du Slot.

Notes : A l'initialisation du MSX, le système cherche dans les Slot la RAM disponible pour chaque plage mémoire et prend la première qu'il trouve sans tenir compte du Memory Mapper, excepté le MSX turbo R qui prend toujours sa RAM interne pour mémoire principale. De plus, les MSX1 n'ont pas de routine pour initialiser les pages du Memory Mapper dans l'ordre 3, 2, 1, 0 sur les plages 0, 1, 2, 3.

Schéma d'une mémoire vive de 128 Ko gérée par le « Memory Mapper » dans le Slot 3-2 :

Huit blocs de 16 Ko soit 128 Ko de mémoire vive
sur Memory Mapper comme sur le modèle VG
8235 MSX2 de Philips



Il est possible de placer une même page sur chacune des plages. Il est également possible de mettre un autre Memory Mapper dans un autre Slot. Dans ce cas, les pages se changeront en même temps dans chacun des Slots.

Attention, la page 0 contient les variables systèmes. Veillez à la laisser toujours dans la plage 3 pour ne pas provoquer un plantage du système.

Utilisation du Memory Mapper :

La gestion du Memory Mapper est particulièrement simple : on utilise les quatre ports d'entrée/sortie, de 0FCh à 0FFh.

En effet, le port 0FCh sert à attribuer la page de Memory Mapper pour la plage 0. Le port 0FDh pour la plage 1, le port 0FEh pour (l'aviez-vous deviné ?) la plage 2 et le port 0FFh pour la plage 3.

0FCh = Page placée sur la plage 0 (00000h~03FFFh)

0FDh = Page placée sur la plage 1 (04000h~07FFFh)

0FEh = Page placée sur la plage 2 (08000h~0BFFFh)

0FFh = Page placée sur la plage 3 (0C000h~0FFFFh)

Ces ports restent accessibles aussi bien en écriture qu'en lecture pour le Memory Mapper interne. En effet, un Memory Mapper en cartouche n'est normalement accessible qu'en lecture. Cependant, il existe des Memory Mapper en cartouche qui gèrent aussi ces ports en lecture. Ces derniers ont été prévus pour les MSX n'ayant pas de pas de Memory Mapper interne. Le standard étant assez floue cette caractéristique, il est devenu déconseillé au développeur d'utiliser ces ports en lecture à cause du conflit provoqué sur les Memory Mapper interne.

A l'initialisation, le Bios du MSX2 (ou plus récent) place les pages de la manière suivante :

mémoire centrale	port I/O	contenu du port	page Memory Mapper
Plage 0 (00000h~03FFFh)	0FCh	003h	page 3
Plage 1 (04000h~07FFFh)	0FDh	002h	page 2
Plage 2 (08000h~0BFFFh)	0FEh	001h	page 1
Plage 3 (0C000h~0FFFFh)	0FFh	000h	page 0

Comme le Memory Mapper est une option sur MSX2, il n'existe aucune variable système ou routine du Bios, le concernant. Si vous désirez utiliser le Memory Mapper, votre programme devra déterminer seul la présence ou l'absence du Memory Mapper ainsi que sa taille mémoire.

Microsoft demande à toute société commercialisant des logiciels MSX2 utilisant le Memory Mapper de porter sur l'emballage la mention « Requires XXX K MEMORY MAPPER ».

Notes :

- Étant donné que le Bios du MSX1 ne gère pas le Memory Mapper, lorsqu'une cartouche de Memory Mapper est insérée, chaque plage mémoire contiendront en général la page 0 au démarrage ou bien, une page indéterminée. Les programmes ayant besoin plus de 16Ko de RAM feront planter le MSX. Il existe des cartouches de Memory Mapper à base de puce programmable qui initialisent les pages dans le bon ordre par eux-même pour éviter ce problème MSX1 mais elles sont peu répandues.
- La Disk-ROM v2.xx, celle du MSX-DOS 2, possède des routines qui permettent de gérer les pages de plusieurs Memory Mapper. Ces routines rendent possible la création d'un gros Ramdisk ainsi que l'utilisation de diverses applications sans risquer un conflit de mémoire. Ces routines sont accessibles via le Bios étendu. (voir le [paragraphe du Bios étendu](#) pour les détails.)

3 Le Bios (*Basic Input/Output System*)

3.1 Introduction au Bios

Le Bios (*Basic Input/Output System*) est un ensemble de routines conçues afin de permettre au programmeur d'accéder au matériel sans que celui soit toujours le même. En effet, le Bios constitue une sorte de tampon entre l'utilisateur et le matériel (« Hardware »). Actuellement, le clavier du MSX est géré par un circuit spécialisé : le PPI 8255. A l'avenir, il se peut qu'un constructeur MSX sorte un modèle avec clavier détaché à liaison infrarouge. Il coule de source que ce nouvel ordinateur ne sera pas muni d'un 8255. Si un programme fait des accès directs au PPI, il ne fonctionnera pas correctement sur le nouveau modèle. Au contraire, si le programme passe par la routine « lire l'état du clavier », il tournera très bien avec le clavier à liaison infrarouge. Le Bios concilie compatibilité des logiciels avec évolution du matériel.

3.2 Table des sauts du Bios en Main-ROM

Les routines du Bios pouvant avoir une taille variable d'un MSX à l'autre, il a donc été nécessaire de créer une table de saut vers chaque routine du Bios afin que tous les logiciels fonctionnent quelque soit le MSX utilisé. Voici la liste de chaque routine avec son adresse mémoire, son nom, sa fonction, les paramètres devant être fournis, les résultats que vous récupèrerez en retour ainsi que les registres qui ont été modifiés par la routine et parfois des remarques sur la routine elle-même, sur son fonctionnement ou les différences entre chaque version de MSX.

00000h (Main-ROM) **STARTUP** (appelée **CHKRAM** dans un premier temps)

Rôle :	Redémarrage du MSX. (Recherche de la Mémoire, initialisation des variables du système, etc.)
Entrée :	Rien.
Sortie :	Rien.
Modifie :	Tout.
Note :	Le MSX va aussi chercher dans chaque Slot de 0-0 à 3-3 la présence de ROM ayant l'entête 41h, 42h et exécuter le programme qu'elle contient.

00004h (Main-ROM) **CGTABL** Character Generator TABLE

Rôle :	Les octets 00004h et 00005h contiennent l'adresse du début table de caractères au code ASCII par défaut.
--------	--

00006h (Main-ROM) **VDP.DR** VDP Data Read port

Rôle : Cet octet renferme l'adresse du port d'entrée du premier port de lecture du processeur vidéo interne. Les logiciels nécessitant des accès vidéo très rapides peuvent adresser directement le VDP (sans passer par le BIOS) à condition d'utiliser cette adresse comme port d'entrée.

Pour plus de précision, voir le chapitre « [Comment accéder au VDP](#) ».

00007h (Main-ROM) **VDP.DW** VDP Data Write port

Rôle : Cet octet renferme l'adresse du port de sortie du premier port d'écriture du processeur vidéo interne. Les logiciels nécessitant des accès vidéo très rapides peuvent adresser directement le VDP (sans passer par le BIOS) à condition d'utiliser cette adresse comme port de sortie.

Pour plus de précision, voir le chapitre « [Comment accéder au VDP](#) ».

00008h (Main-ROM) **SYNCHR** SYNTAX of CHAracter

Rôle : Comparaison du caractère placé sur l'octet suivant le RST 8 avec celui pointé par HL. Si c'est le même, saut à CHRGTR (00010h en Main-ROM) sinon, affichage de « Syntax error ».

Entrée : HL = Adresse du caractère à comparer.

Octet suivant l'instruction RST 8 utilisée pour lancer cette routine = Caractère.

Sortie : HL = Adresse suivante.

A = Caractère pointé par HL.

F = L'indicateur C passe à 1 si le caractère est un chiffre.

l'indicateur Z passe à 1 si le caractère est 00h ou 3Ah (fin d'instruction).

Modifie : AF, HL.

Note : Cette routine est utilisée par l'interpréteur Basic. (RST 8)

0000Ch (Main-ROM) **RDSLt** ReaD SLoT

Rôle : Lecture d'un octet dans un autre Slot.

Entrée : HL = Adresse de l'octet à lire (de 0000h à BFFFh).

A = Numéro du Slot sous la forme binaire F000SSPP.

F doit être à 0 pour Slot primaire ; à 1 pour Slot secondaire.

SS est le numéro de Slot Secondaire.

PP est le numéro de Slot Primaire.

Sortie : A = Octet lu.

Modifie : AF, BC, DE.

Note : Interruptions automatiquement interdites lorsque l'on appelle cette routine mais ne les rétabli pas. Cette routine peut aussi être appelée sous MSX-DOS.

00010h (Main-ROM) **CHRGTR** CHaracter GeTteR

- Rôle : Récupération d'un caractère ou d'un chiffre dans un programme Basic.
- Entrée : HL = Adresse actuelle. (Pointeur)
- Sortie : HL = Adresse de caractère récupéré.
 A = Caractère ou chiffre récupéré.
 F = Indicateur Z à 1 si il s'agit d'un code de fin de ligne (00h ou « : »).
 Indicateur C à 1 si il s'agit d'un chiffre de 0 à 9.
- Modifie : AF, HL.
- Notes : - Cette routine passe les codes d'espacement (020h).
 - Cette routine est utilisée par l'interpréteur Basic. (RST 10)

00014h (Main-ROM) **WRSLT** WRite SLoT

- Rôle : Sélection d'un Slot, puis écriture dans une case mémoire.
- Entrée : HL = Adresse de la case mémoire (de 0000h à BFFFh).
 E = Donnée à écrire.
 A = Numéro du Slot sous la forme F00SSPP.
 F doit être à 0 pour Slot primaire ; à 1 pour Slot secondaire.
 SS est le numéro de Slot Secondaire.
 PP est le numéro de Slot Primaire.
- Sortie : Rien.
- Modifie : AF, BC, D.
- Note : Interruptions automatiquement interdites. Cette routine peut aussi être appelée sous MSX-DOS.

00018h (Main-ROM) **OUTDO** OUT DO

- Rôle : Sortie d'un caractère alphanumérique sur écran, imprimante ou disquette.
- Entrée : A = Code ASCII du caractère à sortir.
 PRTFLG (0F416h) = Indicateur à 1 pour sortie sur imprimante ou disquette.
 PTRFIL (0F864h) = Si différent de 0, alors sortie sur disquette dans le fichier pointé par PTRFIL (0F864h).
- Sortie : Rien.
- Modifie : Rien.
- Notes : - Appel au Hook H.OUTD (0FEE4h).
 - Cette routine est utilisé par l'interpréteur Basic. (RST 18)

0001Ch (Main-ROM) **CALSLT** CALL SLOt

- Rôle : Appel inter-Slot à une adresse.
- Entrée : IX = Adresse à appeler.
 IY = Numéro du Slot sous la forme F000SSPP.
 F doit être à 0 pour Slot primaire ; à 1 pour Slot secondaire.
 SS est le numéro de Slot Secondaire.
 PP est le numéro de Slot Primaire.
- Sortie : Rien.
- Modifie : AF, IX, IY, registres auxiliaires.
- Note : Interruptions automatiquement interdites. Cette routine existe aussi sous MSX-DOS. Ne jamais passer d'arguments à un sous-programme via les registres auxiliaires du Z80 quand vous utilisez CALSTL.

Exemple en assembleur : Appeler une routine du Bios en Main-ROM sous DOS.

```
; Attention, cette routine ne peut appeler SUBROM (0015Ch)

CALSLT EQU 001CH            ; Routine d'appel inter-slot
EXPTBL EQU 0FCC1H          ; Octet indiquant le Slot de la Main-ROM
BEEP    EQU 000C0H          ; Routine d'émission d'un son "Bip"

DEBUT: LD IY, (EXPTBL-1) ; Emplacement de la Main-ROM dans IY
       LD IX, BEEP        ; Adresse de la routine BEEP dans IX
       CALL CALSLT        ; Appel la routine BEEP

END:
```

00020h (Main-ROM) **DCOMPR** Double register COMpare

- Rôle : Comparaison de HL avec DE.
- Entrée : HL = Premier nombre.
 DE = Deuxième nombre.
- Sortie : F = Indicateur C passe à 1 si HL < DE ;
 indicateur Z passe à 1 si HL = DE.
- Modifie : AF.
- Note : Cette routine est utilisé par l'interpréteur Basic. (RST 20)

00024h (Main-ROM) **ENASLT** ENAble Slot

- Rôle : Sélection d'un Slot.
- Entrée : HL = Plage à sélectionner. (0000h, 4000h ou 8000h)
 A = Numéro du Slot sous la forme F000SSPP.
 F doit être à 0 pour Slot primaire ; à 1 pour Slot secondaire.
 SS est le numéro de Slot Secondaire.
 PP est le numéro de Slot Primaire.
- Sortie : Rien.
- Modifie : Tout.
- Notes : - Interruptions automatiquement interdites.
 - Cette routine existe aussi sous MSX-DOS.

Exemple d'utilisation en assembleur : Passage du MSX-DOS au Basic.

```
                  ORG 0D000H
MNROM           EQU 0FCC1H
ENASLT           EQU 00024H
RETURN           EQU 0409BH
DEBUT:           LD A, (MNROM)
                  LD HL, 0
                  CALL ENASLT           ; Bios en page 0
                  LD A, (MNROM)
                  LD HL, 04000H
                  CALL ENASLT           ; Basic page 1
                  JP RETURN            ; Saut à l'interpréteur
                  END DEBUT
```

00028h (Main-ROM) **GETYPR**

- Rôle : Détermination du type de DAC (« Decimal Acumulator »).
- Entrée : VALTYP (F663h) = Type de DAC.
- Sortie : F = S*, C, P/V à 1 et Z à 0 si chiffre entier ;
 S, Z, P/V* à 0 et C à 1 si chiffre simple précision ;
 S, Z, C* à 0 et P/V à 1 si chiffre double précision ;
 C, Z*, P/V à 0 et S à 1 si chaine de caractères.
 * Indicateur sur lequel se baser pour connaître le type.
- Modifie : AF.
- Note : Cette routine est utilisé par l'interpréteur Basic. (RST 28)

0002Bh (Main-ROM) **BASRVN** BASic Rom Version Number

Rôle : Cet octet et le suivant contiennent des renseignements utiles aux programmes destinés à fonctionner dans des pays différents (« international software »).

0002Bh = Les bits 0 à 3 indiquent le types de police de caractères :

0 = japonais ; 1 = Autre.

Les bits 4 à 6 indiquent le format de la date :

0 = AA/MM/JJ ; 1 = MM/JJ/AA ; 2 = JJ/MM/AA

Le bit 7 indique la fréquence des interruptions (VSYNC) :

0 = 60 Hertz ; 1 = 50 Hertz.

0002Ch = Les bits 0 à 3 indiquent le type de clavier :

0 = Japonais ; 1 = Amérique ; 2 = Français,

3 = Anglais ; 4 = Allemand ; 6 = Espagnol.

Les bits 4 à 7 indiquent le type de Basic :

0 = Japonais ; 1 = International.

Note : Les MSX coréens ne respectent pas la norme. Ils indiquent la police de caractères et le type de clavier comme étant japonais.

Tableau des différences par pays :

Pays	Standard TV	Format de la date	Mode d'écran initial	Symbole utilisés selon le type de Basic		
				Longueur de chaine	Remplacement	Monnaie
Angleterre	PAL (50Hz)	JJ/MM/AA	SCREEN0	%	&	£
Allemagne	PAL (50Hz)	JJ/MM/AA	SCREEN0	%	&	\$
Koweit, Egypt	PAL (50Hz)	JJ/MM/AA	SCREEN0/1	%	&	\$
Argentine	PAL (50Hz)	MM/JJ/AA	SCREEN0	%	&	\$
Brésil	PAL (60Hz)	JJ/MM/AA	SCREEN0	%	&	\$
Corée	NTSC (60Hz)	AA/MM/JJ	SCREEN1	%	@	¥
Espagne	PAL (50Hz)	MM/JJ/AA	SCREEN0	%	&	\$
France	SECAM (50Hz)	JJ/MM/AA	SCREEN0	%	&	\$
Japon	NTSC (60Hz)	AA/MM/JJ	SCREEN1	%	@	¥
Pays soviétique	NTSC (60Hz)	MM/JJ/AA	SCREEN0	%	&	\$
USA	NTSC (60Hz)	MM/JJ/AA	SCREEN0	%	&	\$

0002Dh (Main-ROM) **MSXVER** MSX VERsion

Rôle : Version du MSX. Cette octet contient 0 pour un MSX1, 1 pour un MSX2, 2 pour un MSX2+, 3 pour un MSX turbo R.

Note : Les autres numéros sont réservés pour les versions futures de MSX.

0002Eh (Main-ROM)

Rôle : Permet de savoir si l'interface Midi est en interne ou pas.

0002Eh = Le bit 0 est à 1 si le MSX a une interface Midi.

Note : Les autres numéros sont réservés pour les versions futures de MSX.

0002Fh (Main-ROM)

Rôle : Réservé.

00030h (Main-ROM) **CALLF** CALL Far

Rôle : Appel inter-Slot à une adresse.

Entrée : Rien.

Sortie : Rien.

Modifie : AF, IX, IY, registres auxiliaires.

Notes : - CALLF diffère de CALSTL dans la manière d'appeler la routine. Pour appeler CALLF, il faut écrire utiliser l'instruction RST 30 de la façon suivante.

```
RST 30
DB Numéro du Slot (F000SSPP)
DW Adresse à appeler
```

- Interruptions automatiquement désactivées par la routine.

- Cette routine existe aussi sous MSX-DOS.

- Ne jamais passer d'arguments à un sous-programme via les registres auxiliaires du Z80 lorsque vous utilisez CALLF.

00038h (Main-ROM) **KEYINT** Encode KEYboard / timer INTerrupt routine

Rôle : Routine des interruptions masquables.

Entrée : Rien.

Sortie : Rien.

Modifie : Rien.

Notes : - Cette routine appelle les Hooks H.KEYI (0FD9Ah) et H.TIMI (0FD9Fh).

- Appel à la Sub-ROM pour les conversions Roma-Kana.

- Cette routine est appelé à chaque interruption. (RST 38)

0003Bh (Main-ROM) **INITIO** INITIALize Input/Output

Rôle : Initialisation du PSG pour les entrées/sorties.

Entrée : Rien.

Sortie : Rien.

Modifie : Tout.

0003Eh (Main-ROM) **INIFNK** INItialize FuNction Key

Rôle : Ré-initialisation des touches de fonction à leur valeur de départ.
Entrée : Rien.
Sortie : Rien.
Modifie : Tout.

00041h (Main-ROM) **DISSCR** DISable SCReen display

Rôle : Extinction de l'écran.
Entrée : Rien.
Sortie : Rien.
Modifie : AF, BC.

00044h (Main-ROM) **ENASCR** ENABle SCReen display

Rôle : Allumage de l'écran.
Entrée : Rien.
Sortie : Rien.
Modifie : AF, BC.

00047h (Main-ROM) **WRTVDP** WRiTe VDP

Rôle : Écriture dans un registre du VDP.
Entrée : C = Numéro du registre du VDP (0~7 sur MSX1 ; 0~23 et 32~46 sur MSX2 ;
 0~46 sur MSX2+ et Turbo R).
 B = Valeur à écrire.
Sortie : Rien.
Modifie : AF, B.
Note : Appel à la Sub-ROM si le bit EV du registre 0 du VDP est modifié ou si le numéro
 de registre est compris entre 8 et 46.

0004Ah (Main-ROM) **RDVRM** ReaD VRaM

Rôle : Lecture d'une case mémoire de la VRAM (00000h~03FFFh).
Entrée : HL = Adresse de la case mémoire à lire.
Sortie : A = Contenu de la case mémoire lue.
Modifie : AF.
Note : Routine MSX1. Pour lire une case mémoire de VRAM dont l'adresse est
 supérieure à 03FFFh, utilisez la routine NRDVRM.

0004Dh (Main-ROM) **WRTVRM** WRiTe VRaM

- Rôle : Écriture dans une case mémoire de la VRAM.
- Entrée : HL = Adresse de la case mémoire (00000h~03FFFh).
 A = Donnée à écrire.
- Sortie : Rien.
- Modifie : Rien.
- Note : Routine MSX1. Pour écrire un octet dans la VRAM à une adresse supérieure à 03FFFh, utiliser la routine NWRVRM (00177h en Main-ROM).

00050h (Main-ROM) **SETRD** SET address for ReaD

- Rôle : Transfert dans le VDP de l'adresse de la case mémoire où doit s'effectuer la lecture.
- Entrée : HL = Adresse de lecture (00000h~03FFFh).
- Sortie : Rien.
- Modifie : AF.
- Note : Routine MSX1. Pour accéder à une adresse supérieure à 03FFFh, utiliser la routine NSETRD (0016Eh en Main-ROM).

00053h (Main-ROM) **SETWRT** SET address for WRiTe

- Rôle : Transfert dans le VDP de l'adresse de la case mémoire où doit s'effectuer l'écriture.
- Entrée : HL = Adresse d'écriture (00000h~03FFFh).
- Sortie : Rien.
- Modifie : AF.
- Note : Routine MSX1. Pour accéder à une adresse supérieure à 03FFFh, utiliser la routine NSTWRT (00171h en Main-ROM).

00056h (Main-ROM) **FILVRM** FILl VRaM

- Rôle : Remplissage de la mémoire vidéo.
- Entrée : HL = Adresse de début (00000h~03FFFh)
 BC = Longueur.
 A = Donnée.
- Sortie : Rien.
- Modifie : AF, BC.
- Note : Routine MSX1. Pour accéder aux adresses supérieures à 03FFFh, utiliser la routine BIGFIL (0016Bh en Main-ROM).

00059h (Main-ROM) ***LDIRMV*** LoaD Increment Repeat Memory with Vram

Rôle : Transfert de la mémoire vidéo vers la mémoire centrale.
Entrée : HL = Adresse de départ en VRAM (00000h~03FFFh ou 0FFFFh).
 DE = Adresse de destination en RAM centrale (00000h~0FFFFh).
 BC = Longueur du bloc à transférer.
Sortie : Rien.
Modifie : Tout.
Note : Routine MSX1 qui fonctionne dans les modes d'écrans autres que MSX1 mais que sur la page 0.

0005Ch (Main-ROM) ***LDIRVM*** LoaD Increment Repeat Vram with Memory

Rôle : Transfert de la mémoire centrale vers la mémoire vidéo.
Entrée : HL = Adresse de départ en RAM centrale (00000h~0FFFFh).
 DE = Adresse de destination en VRAM (00000h~03FFFh ou 0FFFFh).
 BC = Longueur du bloc à transférer.
Sortie : Rien.
Modifie : Tout.
Note : Routine MSX1 qui fonctionne dans les modes d'écrans autres que MSX1 mais que sur la page 0.

0005Fh (Main-ROM) ***CHGMOD*** CHange MODE

Rôle : Changement de mode d'écran (SCREEN X).
Entrée : A = Mode désiré (0~3 ou 0~8 / 12).
Sortie : SCRMOD (0FCAFh) = Nouveau mode d'écran.
Modifie : Tout.
Note : La palette n'est pas initialisée par cette routine, appelez la routine CHGMDP (001B5h en Sub-ROM) si vous avez besoin de l'initialisation de la palette.

00062h (Main-ROM) **CHGCLR** CHanGe CoLoR

- Rôle : Réactualise les couleurs de l'écran avec les valeurs par défaut.
- Entrée : FORCLR (0F3E9h) = Couleur de texte ou de tracé.
BAKCLR (0F3EAh) = Couleur de fond.
BDRCLR (0F3EBh) = Couleur de bordure.
- Sortie : Rien.
- Modifie : Tout.
- Notes : - En mode graphique, seule la couleur de bordure change.
- Même fonction que CHGCLR (00111h) de la Sub-ROM.

00066h (Main-ROM) **NMI** Non Maskable Interrupt

- Rôle : Routine d'interruptions non-masquables.
- Entrée : Rien.
- Sortie : Rien.
- Modifie : Rien.
- Notes : - Appel au Hook H.NMI (0FDD6h).
- Interruption inutilisée par les MSX.

00069h (Main-ROM) **CLRSR** CLeaR Sprites

- Rôle : Initialisation des Sprites.
- Entrée : Rien.
- Sortie : Rien.
- Modifie : Tout.
- Note : Les valeurs de la table des formes des Sprites sont mises à zéro, les numéros d'affichage de Sprite sont initialisés avec les valeurs de 0 à 31 (si Sprites 16x16) ou 0 à 255 (si 8x8), la couleur des Sprites prend celle défini par FORCLR (0F3E9h) et l'ordonnée des Sprites est réglée à 209 ou 217 selon la hauteur l'écran (192 ou 212 pixels).

0006Ch (Main-ROM) **INITXT** INItialize TeXT mode

- Rôle : Initialisation du mode texte indiqué par la variable OLDSCR (0FCB0h).
- Entrée : Rien.
- Sortie : Rien.
- Modifie : Tout.
- Note : Il s'agit d'une routine MSX1. La palette n'est pas initialisée par cette routine. Si vous désirez initialiser la palette, appelez la routine INIPLT (00141h) de la Sub-ROM juste après celle-ci.

0006Fh (Main-ROM) **INIT32** INItialize Text 32 mode

- Rôle : Initialisation du mode SCREEN 1.
- Entrée : T32NAM (0F3BDh) = Adresse de la table des caractères.
 T32COL (0F3BFh) = Adresse de la table des couleurs.
 T32CGP (0F3C1h) = Adresse de la table des formes.
 T32ATR (0F3C3h) = Adresse de la table des attributs de Sprites.
 T32PAT (0F3C5h) = Adresse de la table des formes de Sprites
- Sortie : Rien.
- Modifie : Tout.
- Notes : - Il n'est pas obligatoire de définir T32NAM, etc car le MSX initialise leur valeur à l'allumage.
 - La palette n'est pas initialisée par cette routine. Si vous désirez initialiser la palette, appelez la routine INIPLT (00141h en Sub-ROM) juste après celle-ci.

00072h (Main-ROM) **INIGRP** INItialize GRaPhic mode

- Rôle : Initialisation du mode SCREEN 2.
- Entrée : GRPNAM (0F3C7h) = Adresse de la table des caractères graphiques.
 GRPCOL (0F3C9h) = Adresse de la table des couleurs.
 GRPCGP (0F3CBh) = Adresse de la table des formes.
 GRPATR (0F3CDh) = Adresse de la table des attributs de Sprites.
 GRPPAT (0F3CFh) = Adresse de la table des formes de Sprites.
- Sortie : Rien.
- Modifie : Tout.
- Notes : - Il n'est pas obligatoire de définir GRPNAM, etc car le MSX initialise leur valeur à l'allumage.
 - La palette n'est pas initialisée par cette routine. Si vous désirez initialiser la palette, appelez la routine INIPLT (00141h) en Sub-ROM juste après celle-ci.

00075h (Main-ROM) **INIMLT** INItialize MuLTicolor mode

- Rôle : Initialisation du mode SCREEN 3.
- Entrée : MLTNAM (0F3D1h) = Adresse de la table des caractères.
MLTCOL (0F3D3h) = Adresse de la table des couleurs.
MLTCGP (0F3D5h) = Adresse de la table des formes.
MLTATR (0F3D7h) = Adresse de la table des attributs de Sprites.
MLTPAT (0F3D9h) = Adresse de la table des formes de Sprites.
- Sortie : Rien.
- Modifie : Tout.
- Notes : - Il n'est pas obligatoire de définir MLTNAM, etc car le MSX initialise leur valeur à l'allumage.
- La palette n'est pas initialisée par cette routine. Si vous désirez initialiser la palette, appelez la routine INIPLT (00141h) en Sub-ROM juste après celle-ci.

00078h (Main-ROM) **SETTXT** SET TeXT mode

- Rôle : Passage direct en mode SCREEN 0 sans initialiser le contenu des tables.
- Entrée : TXTNAM (0F3B3h) = Adresse de la table des caractères.
TXTCGP (0F3B7h) = Adresse de la table des formes.
- Sortie : Rien.
- Modifie : Tout.
- Notes : - Il n'est pas obligatoire de définir TXTNAM et TXTCGP car le MSX initialise leur valeur à l'allumage.
- La palette n'est pas initialisée par cette routine. Si vous désirez initialiser la palette, appelez la routine INIPLT (00141h) en Sub-ROM juste après celle-ci.

0007Bh (Main-ROM) **SETT32** SET Text 32 mode

- Rôle : Passage direct en mode SCREEN 1 sans initialiser le contenu des tables.
- Entrée : T32NAM (0F3BDh) = Adresse de la table des caractères.
T32COL (0F3BFh) = Adresse de la table des couleurs.
T32CGP (0F3C1h) = Adresse de la table des formes.
T32ATR (0F3C3h) = Adresse de la table des attributs de Sprites.
T32PAT (0F3C5h) = Adresse de la table des formes de Sprites.
- Sortie : Rien.
- Modifie : Tout.
- Notes : - Il n'est pas obligatoire de définir T32NAM, etc car le MSX initialise leur valeur à l'allumage.
- La palette n'est pas initialisée par cette routine. Si vous désirez initialiser la palette, appelez la routine INIPLT (00141h) en Sub-ROM juste après celle-ci.

0007Eh (Main-ROM) **SETGRP** SET GRaPhic mode

- Rôle : Passage direct en mode SCREEN 2 sans initialiser le contenu des tables.
- Entrée : GRPNAM (0F3C7h) = Adresse de la table Bitmap.
 GRPCOL (0F3C9h) = Adresse de la table des couleurs.
 GRPCGP (0F3CBh) = Adresse de la table des formes.
 GRPATR (0F3CDh) = Adresse de la table des attributs de Sprites.
 GRPPAT (0F3CFh) = Adresse de la table des formes de Sprites.
- Sortie : Rien.
- Modifie : Tout.
- Notes : - Il n'est pas obligatoire de définir GRPNAM, etc car le MSX initialise leur valeur à l'allumage.
 - La palette n'est pas initialisée par cette routine. Si vous désirez initialiser la palette, appelez la routine INIPLT (00141h) en Sub-ROM juste après celle-ci.

00081h (Main-ROM) **SETMLT** SET MuLTicolor mode

- Rôle : Passage direct en mode SCREEN3 sans initialiser le contenu des tables.
- Entrée : MLTNAM (0F3D1h) = Adresse de la table des caractères.
 MLTCOL (0F3D3h) = Adresse de la table des couleurs.
 MLTCGP (0F3D5h) = Adresse de la table des formes.
 MLTATR (0F3D7h) = Adresse de la table des attributs de Sprites.
 MLTPAT (0F3D9h) = Adresse de la table des formes de Sprites.
- Sortie : Rien.
- Modifie : Tout.
- Notes : - Il n'est pas obligatoire de définir MLTNAM à MLTPAT car le MSX initialise leur valeur à l'allumage.
 - La palette n'est pas initialisée par cette routine. Si vous désirez initialiser la palette, appelez la routine INIPLT (00141h) en Sub-ROM juste après celle-ci.

00084h (Main-ROM) **CALPAT** CALculate PATtern table address

- Rôle : Donne l'adresse du début de la forme d'un Sprite dans la table des formes.
- Entrée : A = Numéro du Sprite.
- Sortie : HL = Adresse cherchée.
- Modifie : AF, DE, HL.

00090h (Main-ROM) **GICINI** GI sound Chip INitalize

Rôle : Initialisation du PSG et des données de l'instruction PLAY.

Entrée : Rien.

Sortie : Rien.

Modifie : Rien.

Notes : - Les interruptions doivent être interdites avant l'appel de cette routine.

 - Après initialisation, les registres auront les valeurs suivantes.

R#0 = 01010101 R#1 = 00000000 R#2 = 00000000

R#3 = 00000000 R#4 = 00000000 R#5 = 00000000

R#6 = 00000000 R#7 = 10111000 R#8 = 00000000

R#9 = 00000000 R#10 = 00000000 R#11 = 00001011

R#12 = 00000000 R#13 = 00000000

00093h (Main-ROM) *WRTPSG* WRiTe PSG

Rôle : Écriture dans les registres du PSG (« Programmable Sound Generator »).

Entrée : A = Numéro de registre (0~15).

E = Donnée à écrire.

Sortie : Rien.

Modifie : Rien.

Notes : - Voici une petite descriptions des registres du PSG.

Registre	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
R#0	8 bits de poids faible de la fréquence de la voix 1							
R#1	-	-	-	-	4 bits forts de la fréquence voix 1			
R#2	8 bits de poids faible de la fréquence de la voix 2							
R#3	-	-	-	-	4 bits forts de la fréquence voix 2			
R#4	8 bits de poids faible de la fréquence de la voix 3							
R#5	-	-	-	-	4 bits forts de la fréquence voix 3			
R#6	-	-	-	Fréquence du générateur de bruit blanc				
R#7	Ports d'E/S du PSG		Désactivation du bruit			Désactivation du son		
	B = 1	A = 0	voix 3	voix 2	voix 1	voix 3	voix 2	voix 1
R#8	-	-	-	V/A	Volume / Amplitude de la voix 1			
R#9	-	-	-	V/A	Volume / Amplitude de la voix 2			
R#10	-	-	-	V/A	Volume / Amplitude de la voix 3			
R#11	8 bits de poids faible de la période de l'enveloppe (T)							
R#12	8 bits de poids fort de la période de l'enveloppe (T)							
R#13	-	-	-	-	Forme de l'enveloppe			
R#14	Port A d'E/S du PSG (à régler toujours en entrée avec le bit 6 du R#7)							
R#15	Port B d'E/S du PSG (à régler toujours en sortie avec le bit 7 du R#7)							

- La valeur du registre 6 se calcul avec la formule suivante :

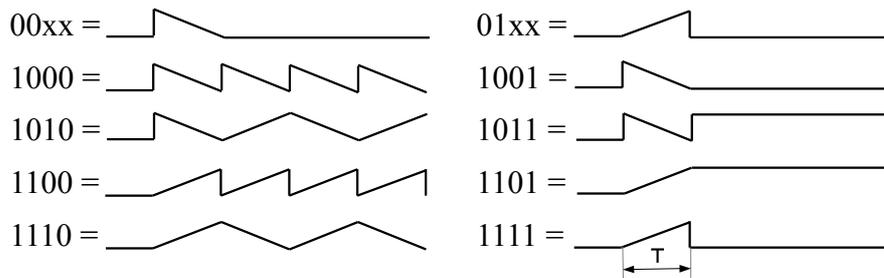
$$\text{Valeur} = 3579545 / (16 \times \text{fréquence du générateur de bruit blanc})$$

- Pour couper le son sur une voix, il faut mettre le volume de la voix à 0 *et* mettre le bit correspondant à cette voix à 1 dans le registre de contrôle (R#7).

- Afin de garantir un bon fonctionnement, le bit 7 du registre 7 doit toujours rester à 1 et le bit 6 à 0 car sur MSX, le port A du PSG fonctionne en entrée et le port B en sortie.

- Le bit 4 des registres 8 à 10 indiquent que le volume est normal (0) ou, contrôlé par le générateur d'enveloppe (1) pour chacune des voix.

- Le registre 13 définit la forme de l'enveloppe.



$$T = (\text{Période de l'enveloppe}) \times 256 / 1,7897725 \text{ (résultat en } \mu\text{s)}$$

Pour les détails : Le bit 0 (Hold) indique le maintien ou pas de l'enveloppe au delà de deux cycles. Le bit 1 (Alternate) indique l'inversion ou pas à chaque nombre paire. Le bit 2 (Attack) indique l'inversion ou pas de l'enveloppe décroissante. Le bit 3 (Continue) indique la continuation au delà de deux cycles.

- Tableau de la note de musique obtenue en fonction de la valeur indiquée aux registres 0~1, 2~3 et 4~5.

Octave \ Note	1	2	3	4	5	6	7	8
Do (C)	D5Dh	6AFh	357h	1ACh	D6h	6Bh	35h	1Bh
Do# (C#)	C9Ch	64Eh	327h	194h	CAh	65h	32h	19h
Ré (D)	BE7h	5F4h	2FAh	17Dh	BEh	5Fh	30h	18h
Ré# (D#)	B3Ch	59Eh	2CFh	168h	B4h	5Ah	2Dh	16h
Mi (E)	A9Bh	54Eh	2A7h	153h	AAh	55h	2Ah	15h
Fa (F)	A02h	501h	281h	140h	A0h	50h	28h	14h
Fa# (F#)	973h	4BAh	25Dh	12Eh	97h	4Ch	26h	13h
Sol (G)	8EBh	476h	23Bh	11Dh	8Fh	47h	24h	12h
Sol# (G#)	86Bh	436h	21Bh	10Dh	87h	43h	22h	11h
La (A)	7F2h	3F9h	1FDh	FEh	7Fh	40h	20h	10h
La# (A#)	780h	3C0h	1E0h	F0h	78h	3Ch	1Eh	0Fh
Si (B)	714h	38Ah	1C5h	E3h	71h	39h	1Ch	0Eh

$$\text{Fréquence} = 1.7897725 / (16 \times \text{la valeur indiquée dans les registres})$$

- Les registres 14 et 15 servent à gérer les ports généraux (manettes de jeu, souris, tablette graphique, etc), la lecture sur cassette et le mode Kana / JIS des MSX japonais.

Registres 14 :

Ce registre permet de lire l'état des broches de 1 à 6 des ports généraux, des claviers japonais et du lecteur cassette à bande.

Bit 0 = Broche 1 du port général sélectionné (Haut si joystick)

Bit 1 = Broche 2 du port général sélectionné (Bas si joystick)

Bit 2 = Broche 3 du port général sélectionné (Gauche si joystick)

Bit 3 = Broche 4 du port général sélectionné (Droite si joystick)

Bit 4 = Broche 6 du port général sélectionné (Bouton 1 si joystick)

Bit 5 = Broche 7 du port général sélectionné (Bouton 2 si joystick)

Bit 6 = 1 pour mode JIS, 0 pour mode Kana (claviers japonais seulement)

Bit 7 = CSAR (Lecture cassette)

Registre 15 :

Ce registre permet de paramétrer l'état des broches de 6 à 8 des ports généraux ainsi que l'état de la LED « code » ou « Kana » selon le type de clavier.

Bit 0 = Broche 6 du port général 1* (1 pour joystick à 2 boutons)

Bit 1 = Broche 7 du port général 1* (1 pour joystick à 2 boutons)

Bit 2 = Broche 6 du port général 2* (1 pour joystick à 2 boutons)

Bit 3 = Broche 7 du port général 2* (1 pour joystick à 2 boutons)

Bit 4 = Broche 8 du port général 1*

Bit 5 = Broche 8 du port général 2*

Bit 6 = Sélection du port général (1 pour port 2)

Bit 7 = État de la LED « code » ou « Kana ». (1 pour éteindre)

* Mettre à 1 si les ports généraux sont utilisés en entrée (lecture).

00096h (Main-ROM) ***RDPSG*** ReaD PSG

Rôle : Lecture d'un registre du PSG.

Entrée : A = Numéro du registre.

Sortie : A = Contenu du registre.

Modifie : Rien.

Note : Voir la routine WRTPSG pour le détail des registres du PSG.

00099h (Main-ROM) ***STRTMS*** STarT background MuSic task

Rôle : Démarre le travail de fond pour PLAY.

Entrée : Rien.

Sortie : Rien.

Modifie : Tout.

0009Ch (Main-ROM) **CHSNS**

Rôle : Mise à jour de l'affichage des touches de fonction d'une part, et vérification du Buffer du clavier d'autre part.

Entrée : Rien.

Sortie : F = L'indicateur Z passe à 0 si il y a un caractère dans le Buffer du clavier.

Modifie : AF.

0009Fh (Main-ROM) **CHGET** CHaracter GET

Rôle : Attente d'un caractère au clavier et retour avec son code.

Entrée : Rien.

Sortie : A = Code ASCII du caractère.

Modifie : AF.

Note : Appel au Hook H.CHGE (0FDC2h).

000A2h (Main-ROM) **CHPUT** CHaracter outPUT

Rôle : Sortie d'un caractère sur écran en mode texte.

Entrée : A = Code du caractère.

Sortie : Rien.

Modifie : Rien.

Note : Appel au Hook H.CHPU (0FDA4h).

000A5h (Main-ROM) **LPTOUT** Line PrinTer OUT

Rôle : Sortie d'un caractère sur imprimante.

Entrée : A = Code du caractère.

Sortie : F = Indicateur C sera à 1 si la routine a été interrompue par un CTRL+STOP.

Modifie : F.

Notes : - Appel au Hook H.LPTO (0FFB6h).
- CTRL+STOP permet de sortir de cette routine en cas de problème avec l'imprimante.

000A8h (Main-ROM) **LPTSTT** Line PrinTer STaTus

Rôle : Vérification de l'état de l'imprimante.

Entrée : Rien.

Sortie : A = 255 si l'imprimante est prête, 0 si elle ne l'est pas.
F = Indicateur Z à si l'imprimante est prête, 1 si elle ne l'est pas.

Modifie : AF.

Note : Appel au Hook H.LPTS (0FFBBh)

000ABh (Main-ROM) *CNVCHR* CoNVert CHaRacter

- Rôle : Teste si en-tête graphique (01) et conversion de caractère.
- Entrée : A = Code du caractère
- Sortie : F = L'indicateur C passe à 0 si le code n'a pas de Header graphique,
l'indicateur Z passe à 1 si le code a été transformé et placé dans A.
- Modifie : AF.
- Note : Si A contient 01 (code de Header graphique) en entrée, GRPHED (0FCA6h) est mis à 1. Le caractère suivant envoyé à la routine sera transformé (masquage du bit 6) à condition que son code soit compris entre 65 et 95.

000AEh (Main-ROM) *PINLIN* Program INput LINE

- Rôle : Place le curseur à la ligne suivante puis stocke les caractères entrés au clavier dans le Buffer jusqu'à ce que la touche STOP ou RET soit pressés.
- Entrée : Rien.
- Sortie : HL = Adresse du premier octet dans le Buffer -1.
F = Indicateur C mis à 1 si c'est la touche STOP qui a été pressée.
- Modifie : Tout.
- Notes : - Appel au Hook H.PINL (0FDDBh)
- Si AUTFLG (0F6AAh) est à 1, appellera la routine INLIN (000B1h) à la place.

000B1h (Main-ROM) *INLIN* INput LINE

- Rôle : Stocke les caractères entrés au clavier dans le Buffer jusqu'à ce que la touche STOP ou RET soit pressés.
- Entrée : Rien.
- Sortie : HL = Adresse du premier octet dans le Buffer -1.
F = Indicateur C mis à 1 si c'est la touche STOP qui a été pressée.
AUTFLG (0F6AAh) = 1.
- Modifie : Tout.
- Notes : - Appel au Hook H.INLI (0FDE5h).
- Par rapport à PINLIN, cette routine termine la ligne précédente (voir LINTTB (0FBB2h)).

000B4h (Main-ROM) ***QINLIN*** Questiou mark and INput LIne

Rôle : Idem INLIN sauf que cette routine affiche un point d'interrogation.
Entrée : Rien.
Sortie : HL = Adresse du premier octet dans le Buffer -1.
 F = Indicateur C à 1 si un ^C a eu lieu.
Modifie : Tout.
Note : Appel au Hook H.QINL (0FDE0h).

000B7h (Main-ROM) ***BREAKX*** BREAK X

Rôle : Test du CTRL+STOP
Entrée : Rien.
Sortie : F = Indicateur C sera à 1 si ^C en cours.
Modifie : AF
Exemple d'utilisation :

```
LOOP: CALL BREAKX
      JR  NC, LOOP
      RET
```

000BAh (Main-ROM) ***ISCNTC*** IS CoNtrol C ?

Rôle : Test de la touche STOP ignoré par l'interpréteur Basic.
Entrée : BASROM (0FBB1h) = 1 si le programme Basic est en cartouche (touche STOP ignorée).
Sortie : Rien.
Modifie : Rien.
Note : Cette routine est utilisé par l'interpréteur Basic.

000BDh (Main-ROM) ***CKCNTC*** ChecK CoNtrol C

Rôle : Test de la touche STOP ignoré par l'interpréteur Basic.
Entrée : BASROM (0FBB1h) = 1 si programme en cartouche (touche STOP ignorée).
Sortie : Rien.
Modifie : Rien.
Notes : - Cette routine est utilisé par l'interpréteur Basic.
 - Par rapport à ISCNTC, cette routine met le pointeur de texte du Basic TEMPPT (0F678h) à 0 afin d'interdire la reprise de l'exécution du programme avec CONT (« CONTinue »).

000C0h (Main-ROM) **BEEP** BEEP buzzer

Rôle : Émission d'un bref « bip » sonore.
Entrée : Rien.
Sortie : Rien.
Modifie : Tout.
Note : Appel à la Sub-ROM.

000C3h (Main-ROM) **CLS** CLear Screen

Rôle : Effacement de l'écran dans tous les modes.
Entrée : F = Indicateur Z à 1.
Sortie : Rien.
Modifie : AF, BC.
Note : Appel à la Sub-ROM.

000C6h (Main-ROM) **POSIT** POSITion cursor

Rôle : Modification des coordonnées du curseur
Entrée : H = Numéro de colonne
 L = Numéro de ligne
Sortie : Rien.
Modifie : AF.
Note : Bien que n'ayant pas été modifiée sur MSX2, cette routine fonctionne parfaitement en mode 80 colonnes.

000C9h (Main-ROM) **FNKSB** FuNction Key diSplay enaBled

Rôle : Affichage des touches de fonction si DSPFNK est activé.
Entrée : CNSDFG (0F3DEh) = 1 pour activer la routine DSPFNK (000CFh en Main-ROM) et l'appeler ; 0 pour désactiver DSPFNK.
Sortie : Rien.
Modifie : Tout.
Note : Appel au Hook H.DSPF (0FDB3h).

000CCh (Main-ROM) **ERAFNK** ERAse FuNction Key

Rôle : Supprime l'affichage des touches de fonction.
Entrée : Rien.
Sortie : Rien.
Modifie : Tout.
Note : Appel au Hook H.ERAF (0FDB8h).

000CFh (Main-ROM) **DSPFNK** DiSPlay FuNction Key

- Rôle : Affichage des touches de fonction en mode texte.
Entrée : Rien.
Sortie : Rien.
Modifie : Tout.
Note : Appel au Hook H.DSPF (0FDB3h).

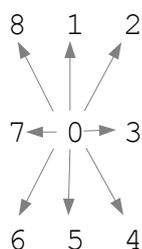
000D2h (Main-ROM) **TOTEXT** force screen TO TEXT mode

- Rôle : Retour au mode texte précédant le passage en mode graphique.
Entrée : OLDSCR (0FCB0h) = Mode texte dans lequel nous étions avant le passage en mode graphique. Nous retournons dans ce mode.
Sortie : Rien.
Modifie : Tout.
Notes : - Appel au Hook H.TOTE (0FDBDh).
 - Cette routine fait appel à CHGMDP (001B5h en Sub-ROM) sur MSX2 ou plus récent.

000D5h (Main-ROM) **GTSTCK** GeT joySTiCK status

- Rôle : Renvoi de la direction actuelle de la manette.
Entrée : A = Numéro de manette (0 ~ 2).
 0 = Touches du curseur du clavier.
 1 = manette 1.
 2 = manette 2.
Sortie : A = Direction de manette (0~8).

Format :



Lors de l'utilisation du clavier comme joystick 0, il faut appuyer simultanément sur deux flèches afin d'obtenir une diagonale. Par exemple, les touches **BAS** et **DROITE** donnent 4 lorsqu'elles sont pressées.

- Modifie : Tout.

000D8h (Main-ROM) **GTTRIG** GeT TRIGger button status

Rôle : Renvoi de l'état du bouton de manette sélectionné.

Entrée : A = Numéro du bouton (0~4).

 0 = Barre d'espacement du clavier ;

 1 = Bouton A de la manette 1 ;

 2 = Bouton A de la manette 2 ;

 3 = Bouton B de la manette 1 ;

 4 = Bouton B de la manette 2.

Sortie : A = 255 si le bouton est enfoncé, 0 si ce n'est pas le cas.

Modifie : AF.

000DBh (Main-ROM) **GTPAD** GeT touch PAD

- Rôle : Lecture de la tablette graphique.
 Entrée : A = Numéro de l'opération à effectuer (0~7 sur MSX1 ou 0~19 sur les autres).
 Sortie : A = Valeur résultante.
 Modifie : Tout.
 Notes : - La tablette graphique doit être une NEC PC-6051 ou compatible.
 - Voici la liste des opérations possibles :

Entrée	Port	Valeur résultante
0	Général n°1	0FFh si le stylet est en contacte avec la tablette graphique, sinon 0.
1		Abscisse de la tablette
2		Ordonnée de la tablette
3		0FFh si le bouton du stylet est pressé, sinon 0.
4	Général n°2	0FFh si le stylet est en contacte avec la tablette graphique, sinon 0.
5		Abscisse de la tablette
6		Ordonnée de la tablette
7		0FFh si le bouton du stylet est pressé, sinon 0.
8	Crayon optique	0FFh si le crayon optique touche l'écran, sinon 0.
9		Abscisse du crayon optique.
10		Ordonnée du crayon optique.
11		0FFh si le bouton du crayon optique est pressé, sinon 0.
12	Général n°1	Toujours à 0FFh.
13		Abscisse de la souris ou du Track-Ball.
14		Ordonnée de la souris ou du Track-Ball.
15	Général n°2	Toujours à 0. (inutilisé)
16		Toujours à 0FFh.
17		Abscisse de la souris ou du Track-Ball.
18		Ordonnée de la souris ou du Track-Ball.
19		Toujours à 0. (inutilisé)

- Les valeurs données par le crayon optique ne sont valables que lorsque le crayon optique touche l'écran.
- Effectuer les opérations 12 ou 16 juste avant d'effectuer une lecture des coordonnées d'une souris ou d'un Track-Ball sinon, les valeurs obtenues ne seront pas garanties correctes.
- Pour tester l'état des boutons de la souris ou du Track-Ball, veuillez utiliser la routine GTTRIG (000DBh en Main-ROM).
- Le crayon optique est pas géré par le MSX turbo R. (A = 0 en sortie.)

000DEh (Main-ROM) **GTPDL** GeT PaDdLe

Rôle : Lecture d'une molette (« Paddle Controller ») ASCII.

Entrée : A = Paramètre d'entrée (1~12).

	Port général 1	Port général 2
Molette A	1	2
Molette B	3	4
Molette C	5	6
Molette D	7	8
Molette E	9	10
Molette F	11	12

Sortie : A = Valeur en fonction de l'angle de rotation (0~255).

Modifie : Tout.

Notes :
- Les molettes du jeu Arkanoïd ne sont pas prises en charge par cette routine.
- Cette routine n'a pas d'effet sur MSX turbo R. (A = 0 en sortie)

000E1h (Main-ROM) **TAPION** TAPe In ON

- Rôle : Démarrage du moteur du magnétophone et lecture d'un signal d'en-tête ou d'un signal séparateur.
- Entrée : Rien.
- Sortie : F = Indicateur C à 1 si opération interrompue (après une erreur par exemple).
- Modifie : Tout.
- Notes : - Voir TAPOON (000EAh en Main-ROM) pour les explications du signal d'en-tête et du signal séparateur.
- La vitesse (1200 ou 2400 bauds) est déterminée lors de la lecture du signal.
- Cette routine n'a pas d'effet sur MSX turbo R. (l'indicateur C = 1 en sortie)

000E4h (Main-ROM) **TAPIN** TAPe IN

- Rôle : Lecture d'un octet depuis la cassette.
- Entrée : Rien.
- Sortie : A = Octet lu.
- F = Indicateur C à 1 si opération interrompue.
- Modifie : Tout.
- Note : Cette routine n'a pas d'effet sur MSX turbo R. (l'indicateur C = 1 en sortie)

000E7h (Main-ROM) **TAPIOF** TAPe In OFF

- Rôle : Fin de lecture, arrêt moteur.
- Entrée : Rien.
- Sortie : Rien.
- Modifie : Rien.
- Note : Cette routine n'a pas d'effet sur MSX turbo R.

000EAh (Main-ROM) **TAPOON** TAPE Out ON

- Rôle : Démarrage du moteur du magnétophone et écriture d'un signal d'en-tête ou d'un signal séparateur.
- Entrée : A = 0 pour écrire un signal un séparateur, 1 pour un signal d'en-tête.
- Sortie : F = Indicateur C à 1 si l'opération a été interrompue (après un CTRL+STOP par exemple).
- Modifie : Tout.
- Notes : - Le signal d'en-tête sert à marquer le début d'un fichier alors que l'autre signal sert à séparer les différentes parties d'un fichier.

Le MSX reconnaît trois types de fichiers sur cassettes : Les fichiers Basic, les fichiers de texte ASCII et les binaires. Voici le détail de chacun d'eux :

Format d'un fichier Basic :

Début du fichier	Signal d'en-tête (de 6.7 sec)							
En-tête du fichier	0D3h	0D3h	0D3h	0D3h	0D3h	0D3h	0D3h	0D3h
	0D3h	0D3h	Nom du fichier (6 octets)					
	Espace vide de durée quelconque							
	Signal séparateur (de 1.7 sec)							
.	Début...							
.	Programme Basic condensé							
.	...fin	000h	000h	000h	000h	000h	000h	000h
Fin du fichier	000h	000h						

Le fichier Basic commence par un signal d'en-tête suivi de dix octets 0D3h pour indiquer qu'il contient un programme Basic condensé et du nom du fichier. Après cette en-tête, il doit y avoir un séparateur qui annonce l'emplacement du programme. Le fichier se fini par une suite de sept 000h.

Format d'un fichier binaire :

Début du fichier	Signal d'en-tête (de 6.7 sec)							
En-tête du fichier	0D0h	0D0h	0D0h	0D0h	0D0h	0D0h	0D0h	0D0h
	0D0h	0D0h	Nom du fichier (6 octets)					
	Espace vide de durée quelconque							
	Signal séparateur (de 1.7 sec)							
	Adrs. début		Adrs. de fin		Adrs. Execut.		Début...	
.	Données binaires (Programme machine ou autre)							
.								
.								
Fin du fichier	...fin							

Le fichier binaire commence par un signal d'en-tête suivi de dix octets 0D0h indiquant qu'il s'agit d'un programme machine et du nom du fichier. Après cette en-tête, il doit y avoir un séparateur qui annonce l'adresse de début du programme, l'adresse de fin, l'adresse d'exécution. Ensuite, vient les données.

Format d'un fichier ASCII :

Début du fichier	Signal d'en-tête (de 6.7 sec)							
En-tête du fichier	0EAh	0EAh	0EAh	0EAh	0EAh	0EAh	0EAh	0EAh
	0EAh	0EAh	Nom du fichier (6 octets)					
	Espace vide de durée quelconque							
	Signal séparateur (de 1.7 sec)							
.	Début...							
	256 caractères de texte ASCII							
	Signal séparateur (de 1.7 sec)							
	256 caractères de texte ASCII							
	.							
	Signal séparateur (de 1.7 sec)							
Fin du fichier	Reste des caractères							...fin

Le fichier ASCII commence par un signal d'en-tête suivi de dix octets 0EAh pour indiquer qu'il contient de texte ASCII puis, du nom du fichier. Après cette en-tête, il doit y avoir un séparateur qui annonce l'emplacement de 256 caractères de texte ASCII. Si le texte dépasse les 256 caractères, il devra y avoir un signal séparateur entre chaque 256 caractères.

- Cette routine n'a pas d'effet sur MSX turbo R. (l'indicateur C = 1 en sortie)

000EDh (Main-ROM) **TAPOUT** TAPe OUT

Rôle : Écriture d'un octet sur la cassette.

Entrée : A = Octet à écrire.

Sortie : F = Indicateur C à 1 si l'opération a été interrompue. (toujours à 1 sur MSX Turbo R.)

Modifie : Tout.

Note : Cette routine n'a pas d'effet sur MSX turbo R. (l'indicateur C = 1 en sortie)

000F0h (Main-ROM) **TAPOOF** TAPe Out OFF

Rôle : Fin d'écriture, arrêt du moteur.

Entrée : Rien.

Sortie : Rien.

Modifie : Rien.

000F3h (Main-ROM) **STMOTR** SeT MOTor

Rôle : Changement d'état du moteur du magnétophone.
Entrée : A = 0 pour arrêter le moteur.
 1 pour démarrer le moteur.
 255 pour inverser l'état actuel.
Sortie : Rien.
Modifie : AF.
Note : Cette routine n'a pas d'effet sur MSX turbo R.

000F6h (Main-ROM) **LFTQ** LeFT in Queue

Rôle : Donne le nombre d'octets restants dans une queue.
Entrée : A = Numéro de queue.
Sortie : A = Nombre d'octets.
Modifie : AF, BC, HL.

000F9h (Main-ROM) **PUTQ** PUT in Queue

Rôle : Empilage d'un octet sur une queue.
Entrée : A = Numéro de queue.
 E = Donnée à mettre.
Sortie : F = Indicateur zéro à 1 si la queue est entièrement remplie.
Modifie : AF, BC, HL.

000FCh (Main-ROM) **RIGHTC** RIGHT Current

- Rôle : Déplacement du curseur graphique d'un pixel vers la droite.
Entrée : Rien.
Sortie : Rien.
Modifie : AF, CLOC (0F92Ah) et éventuellement CMASK (0F92Ch).
Note : Appel à la Sub-ROM si le mode d'écran est supérieur au SCREEN 4.

Exemple en Basic :

```
10 SCREEN 2
20 PSET (0,0),1
30 X=PEEK(&HF92A)+PEEK(&HF92B)*256:Y=PEEK(&HF92C)
40 DEFUSR=&HFC:A=USR(0) ' Appelle RIGHTC
40 X=PEEK(&HF92A)+PEEK(&HF92B)*256:Y=PEEK(&HF92C)
50 SCREEN 0
60 PRINT"X=";X,"Y=";Y,"X2=";X2,"Y2=";Y2
```

Ce programme affiche : « X=0 Y=128 X2=0 Y2=64 »
(en binaire, 128 = 10000000B et 64 = 01000000B)

000FFh (Main-ROM) **LEFTC** LEFT Current

- Rôle : Déplacement du curseur graphique d'un pixel vers la gauche.
Entrée : Rien.
Sortie : Rien.
Modifie : AF, CLOC (0F92Ah) et éventuellement CMASK (0F92Ch).
Notes : - Passe à la fin de la ligne précédente sur l'abscisse du curseur était 0.
- Appel à la Sub-ROM si le mode d'écran est supérieur au SCREEN 4.

00102h (Main-ROM) **UPC** UP Current

- Rôle : Déplacement du curseur graphique d'un pixel vers le haut.
Entrée : Rien.
Sortie : Rien.
Modifie : AF et éventuellement CLOC (0F92Ah).
Note : Appel à la Sub-ROM si le mode d'écran est supérieur au SCREEN 4.

00105h (Main-ROM) **TUPC** Test and UP Current

Rôle : Déplacement du curseur d'un pixel vers le haut avec test de dépassement d'écran.
Entrée : Rien.
Sortie : F = L'indicateur C passe à 1 si le curseur n'a pu être déplacé (sortie d'écran).
Modifie : AF et éventuellement CLOC (0F92Ah).
Note : Appel à la Sub-ROM si le mode d'écran est supérieur au Screen 5.

00108h (Main-ROM) **DOWNC** DOWN Current

Rôle : Déplacement du curseur graphique d'un pixel vers le bas.
Entrée : Rien.
Sortie : Rien.
Modifie : AF et éventuellement CLOC (0F92Ah).
Note : Appel à la Sub-ROM si le mode d'écran est supérieur au SCREEN 4.

0010Bh (Main-ROM) **TDOWNC** Test and DOWN Current

Rôle : Déplacement du curseur graphique d'un pixel vers le bas avec test de dépassement d'écran.
Entrée : Rien.
Sortie : F = Indicateur C à 1 si si le curseur n'a pu être déplacé (sortie d'écran).
Modifie : AF et éventuellement CLOC (0F92Ah).
Note : Appel à la Sub-ROM si le mode d'écran est supérieur au SCREEN 4.

0010Eh (Main-ROM) **SCALXY** SCALe X & Y

Rôle : Vérifie si les coordonnées contenues dans BC et DE dépassent ou pas de l'écran. Les coordonnées seront ajustées en cas de dépassement.
Entrée : BC = Abscisse (0 ~ 0FFFFh).
 DE = Ordonnée (0 ~ 0FFFFh).
Sortie : BC = Abscisse valide.
 DE = Ordonnée valide.
 F = Indicateur C à 0 si BC ou DE était hors de l'écran.
Modifie : AF.
Notes : - Lorsque l'une des deux coordonnées est négative, cette routine prend 0 comme nouvelle coordonnée. De même, si une coordonnée dépasse sa valeur maximale autorisée (255 ou 511 pour X, 191 ou 211 pour Y), la routine prend la valeur maximale autorisée comme nouvelle coordonnée.
 - Appel à la Sub-ROM si le mode d'écran est supérieur au SCREEN 4.

00111h (Main-ROM) **MAPXYC** MAP X&Y to Current

Rôle : Convertie les coordonnées graphiques contenues dans BC et DE en une adresse mémoire vidéo et un masque. (Pas utile dans les SCREEN 5 à 8).

Entrée : BC = Abscisse.
DE = Ordonnée.

Sortie : CLOC (0F92Ah) = Dans les SCREEN 2 à 4 et 10 à 12, adresse en VRAM correspondante aux coordonnées. Dans les SCREEN 5 à 8, abscisse.
CMASK (0F92Ch) = Dans les SCREEN 2 à 4 et 10 à 12, masque à appliquer. Dans les SCREEN 5 à 8, ordonnée.

Note : Appel à la Sub-ROM si le mode d'écran est supérieur au SCREEN 4.

00114h (Main-ROM) **FETCHC** FETCH Current

Rôle : Récupération de l'adresse VRAM actuelle et du masque actuel (ou des coordonnées du pixel pour les SCREEN 5 à 8).

Entrée : Rien.

Sortie : HL = Dans les SCREEN 2 à 4, adresse VRAM du curseur graphique.
Dans les SCREEN 5 à 8, abscisse du pixel.
A = Dans les SCREEN 2 à 4, masque du curseur graphique.
Dans les SCREEN 5 à 8, ordonnée du pixel.
Modifie : F.

00117h (Main-ROM) **STOREC** STORE Current

Rôle : Sauvegarde de l'adresse VRAM actuelle et du masque actuel (ou des coordonnées).

Entrée : HL = Dans les SCREEN 2 à 4, adresse VRAM du curseur graphique.
Dans les SCREEN 5 à 8, abscisse du pixel.
A = Dans les SCREEN 2 à 4, masque du curseur graphique.
Dans les SCREEN 5 à 8, ordonnée du pixel.

Sortie : Rien.

Modifie : Rien.

00123h (Main-ROM) ***NSETCX*** Next SET Current X

Rôle : Affichage de plusieurs pixels à partir du curseur graphique actuel.

Entrée : HL = Nombre de pixels à afficher.

 CLOC (0F92Ah) = Dans les SCREEN 2 à 4 et 10 à 12, adresse où se trouve le curseur graphique en VRAM. Dans les SCREEN 5 à 8, abscisse actuelle du curseur graphique.

 CMASK (0F92Ch) = Dans les SCREEN 2 à 4 et 10 à 12, masque à appliquer. Dans les SCREEN 5 à 8, ordonnée actuelle du curseur graphique.

 ATRBYT (0F3F2h) = Couleur du pixel (0~3 en SCREEN 6 ; 0~15 dans les SCREEN 5 et 7 ; 0~255 dans les SCREEN 8 à 12).

Sortie : Rien.

Modifie : Tout.

00126h (Main-ROM) ***GTASPC*** GeT ASPeCt ratio

Rôle : Renvoi de l'aspect du cercle.

Entrée : Rien.

Sortie : DE = Contenu de ASPCT1 (0F40Bh)

 HL = Contenu de ASPCT2 (0F40Dh)

Modifie : Rien.

00129h (Main-ROM) ***PNTINI*** PaiNT INItialize

Rôle : Initialisation de la procédure de peinture.

Entrée : Rien.

Sortie : Rien.

Modifie : AF.

0012Ch (Main-ROM) ***SCANR*** SCAN Right

Rôle : Recherche vers la droite du premier pixel d'une autre couleur

Entrée : DE = Nombre de pixels sur lesquels la recherche doit s'effectuer

Sortie : DE = Position du pixel recherché

Modifie : Tout.

Notes : - Un exemple, si vous chargez DE avec 100h et que le premier pixel d'une autre couleur se trouve à 4 pixels, la routine renverra 0FCh dans DE

 - Appel à la Sub-ROM si le mode d'écran est supérieur au SCREEN 4.

0012Fh (Main-ROM) **SCANL** SCAN Left

- Rôle : Recherche vers la gauche du premier pixel d'une autre couleur.
- Entrée : DE = Nombre de pixels sur lesquels la recherche doit s'effectuer.
- Sortie : DE = Position du pixel recherché.
- Modifie : Tout.
- Notes : - Un exemple, si vous chargez DE avec 120h et que le premier pixel d'une autre couleur se trouve à 9 pixels, la routine renverra 117h dans DE
- Appel à la Sub-ROM si le mode d'écran est supérieur au SCREEN 4.

00132h (Main-ROM) **CHGCAP** CHanGe CAPs lamp status

- Rôle : Allume ou éteint le voyant CAPS LOCK.
- Entrée : A = 0 pour éteindre le voyant ; 1 pour l'allumer.
- Sortie : Rien.
- Modifie : AF.

00135h (Main-ROM) **CHGSND** CHanGe SouND port status

- Rôle : Mettre ou couper le son.
- Entrée : A = 0 pour couper : 1 pour mettre le son.
- Sortie : Rien.
- Modifie : AF.

00138h (Main-ROM) **RSLREG** Read primary Slot REGISTER

Rôle : Lecture du registre des Slots primaires.

Entrée : Rien.

Sortie : A = Valeur du registre.

Modifie : Rien.

Note : Format du registre :

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Slot de la plage 3 (C000h~FFFFh)	Slot de la plage 2 (8000h~BFFFh)	Slot de la plage 1 (4000h~7FFFh)	Slot de la plage 0 (0000h~3FFFh)				

0013Bh (Main-ROM) **WSLREG** Write Slot REGISTER

Rôle : Écriture dans le registre des Slots primaires.

Entrée : A = Donnée à écrire.

Sortie : Rien.

Modifie : Rien.

Note : Format du registre :

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Slot de la plage 3 (C000h~FFFFh)	Slot de la plage 2 (8000h~BFFFh)	Slot de la plage 1 (4000h~7FFFh)	Slot de la plage 0 (0000h~3FFFh)				

0013Eh (Main-ROM) **RDVDP** Read VDP status register

Rôle : Lecture du registre de statut du VDP.

Entrée : Rien.

Sortie : A = Contenu du registre.

Modifie : Rien.

Note : Sur V9938 ou plus récent, cette routine lit le registre défini par le registre 15.

00141h (Main-ROM) **SNSMAT** Scan Specified row in keyboard MATRIX

Rôle : Lecture d'une ligne dans la matrice clavier

Entrée : A = Numéro de la ligne (0~8 ou 0~10 pour les claviers avec pavé numérique)

Sortie : A = état de la ligne, les bits correspondants aux touches enfoncées sont à 0

Modifie : AF, C

Note : Les matrices de clavier varient d'un MSX à l'autre. Veuillez voir les [exemples donnés dans l'annexe](#) pour vous faire une idée.

00144h (Main-ROM) **PHYDIO** PHYsical Disk I/O

- Rôle : Lecture ou écriture de secteur(s) sur un disque logique .
- Entrée : A = Numéro du disque logique (0 ~ 7).
 B = Nombre de secteur.
 C = Identifiant du type de média utilisé. (0F8h~0FFh).
 0F8h pour les disquette 1DD, 9 secteurs par pistes ;
 0F9h pour les disquette 2DD, 9 secteurs par pistes ;
 0FAh pour les disquette 1DD, 9 secteurs par pistes ;
 0FBh pour les disquette 2DD, 9 secteurs par pistes ;
 0FCh pour les disquette 2D, 8 secteurs par pistes ;
 0FDh pour les disquette 2D, 8 secteurs par pistes ;
 0FEh pour les disquette 1D, 8 secteurs par pistes ;
 0FFh pour les disquette 1D, 8 secteurs par pistes.
- DE = Numéro du premier secteur.
 HL = Adresse de destination en mémoire vive.
 F = Indicateur C à 1 pour écrire, 0 pour lire.
- Sortie : F = Indicateur C à 1 si erreur rencontrée, sinon 0.
 A = Code d'erreur.
 B = Nombre de secteur non lus.
- Modifie : Tout.
- Notes : - Appel au Hook H.PHYD (0FFA7h).
 - Aucun effet si il n'a y pas de Disk-ROM (sauf l'appel au Hook H.PHYD).
- Exemple : Lire le secteur 0 d'une disquette double face 3.5" en assembleur.

```
PHYDIO    EQU 0114H
          ORG 0D000H
Debut:    LD A,0                    ; Lecteur "A:"
          LD B,1                ; nombre de secteur à lire = 1
          LD C,0F9H            ; Disquette double face 3.5"
          LD DE,0              ; premier secteur
          LD HL,(0F351h)      ; vers Buffer de secteur du Disk-Basic
          OR A                 ; Lecture (mettre SCF pour écriture)
          CALL PHYDIO         ; Lecture du secteur 0
          END Debut
```

00147h (Main-ROM) **FORMAT** disk FORMATter

Rôle : Formater une disquette.

Entrée : Rien.

Sortie : Rien.

Modifie : Tout.

Notes : - Appel au Hook H.FORM (0FFACh).
 - Aucun effet si il n'a y pas de Disk-ROM (sauf l'appel au Hook H.FORM).

0014Ah (Main-ROM) **ISFLIO** IS FiLe I/O

Rôle : Vérification du fonctionnement des lecteurs.

Entrée : Rien.

Sortie : A = 0 si accès en cours.

Modifie : AF.

Notes : - Appel au Hook H.ISFL (0FEDFh).
 - Aucun effet si il n'a y pas de Disk-ROM (sauf l'appel au Hook H.ISFL).

0014Dh (Main-ROM) **OUTDLP** OUT Do Line Printer

Rôle : Sortie d'un caractère sur imprimante.

Entrée : A = Code ASCII du caractère à sortir.

Sortie : Rien.

Modifie : F.

Note : Par rapport à LPTOUT, cette routine transforme les TAB en espaces, transforme les caractères graphiques lors de l'utilisation d'une imprimante non-MSX, et saute à « device I/O error » lors d'un CTRL+STOP.

00150h (Main-ROM) **GETVCP** GET VoiCe Pointer

Rôle : Obtention du pointeur pour une queue musicale.

Entrée : A = Numéro de voix (1 ~ 3).

Sortie : HL = Pointeur de la voix désiré.

Modifie : AF.

Note : Utilisé uniquement pour jouer de la musique en arrière-plan.

00153h (Main-ROM) **GETVC2** GET VoiCe 2

- Rôle : Renvoie le pointeur à VOICEN (0FB38h).
Entrée : L = Pointeur de l'instruction PLAY.
Sortie : HL = Nouvelle valeur du pointeur.
Modifie : AF.
Note : Utilisé uniquement pour jouer de la musique en arrière-plan.

00156h (Main-ROM) **KILBUF** KILl Buffer

- Rôle : Vide le Buffer du clavier.
Entrée : Rien.
Sortie : Rien.
Modifie : HL.

00159h (Main-ROM) **CALBAS** CALl BASic

- Rôle : Appel à des sous-programmes dans la ROM Basic depuis un autre Slot (appel inter-Slot)
Entrée : IX = Adresse à appeler.
Sortie : Rien.
Modifie : AF, IX, IY, registres auxiliaires.

Routines ajoutées aux MSX2 :

0015Ch (Main-ROM) **SUBROM** call SUB-ROM

- Rôle : Appel à une adresse dans la Sub-ROM.
Entrée : IX = Adresse à appeler.
Sortie : Rien.
Modifie : Tout sauf IX, IY et registres auxiliaires du Z80.
Notes : - Routine absente sur MSX1.
 - Méthode pour utiliser cette routine :

```
SUBROM    EQU 015CH  
  
EXPL:     PUSH IX  
          LD IX,adresse    ; routine de la Sub-ROM  
          JP SUBROM
```

Une fois la routine de la Sub-ROM exécutée, on aura un retour à la suite du programme EXPL (derrière le JP SUBROM). Attention, cette routine ne fonctionne pas sous MSX-DOS!

0015Fh (Main-ROM) **EXTROM** call EXTernal ROM

Rôle : Appel à une adresse dans une ROM externe.
Entrée : IX = Adresse à appeler.
Sortie : Rien.
Modifie : Tout sauf IY et les registres auxiliaires du Z80.
Note : Routine absente sur MSX1.

00162h (Main-ROM) **CHKSLZ** CHecK Slot Z

Rôle : Recherche du Slot de la Sub-ROM.
Entrée : Rien.
Sortie : Rien.
Modifie : Tout.
Note : Routine absente sur MSX1.

00165h (Main-ROM) **CHKNEW** CHecK NEW screen mode

Rôle : Vérification du SCREEN.
Entrée : Rien.
Sortie : F = Indicateur C à 0 si le mode d'écran actuel est entre SCREEN 5 et 12.
Modifie : AF.
Note : Routine absente sur MSX1.

00168h (Main-ROM) **EOL** erase to End Of Line

Rôle : Effacement jusqu'à la fin de la ligne (équivalent au CTRL E du Basic).
Entrée : H = Colonne à partir de la quelle il faut effacer.
 L = ligne à effacer.
Sortie : Rien.
Modifie : Tout.
Note : Routine absente sur MSX1.

0016Bh (Main-ROM) **BIGFIL** BIG FIL1

Rôle : Remplissage de la mémoire vidéo.
Entrée : HL = Adresse de début (00000h~0FFFFh).
 BC = Longueur.
 A = Donnée.
Sortie : Rien.
Modifie : AF, BC.
Note : Routine absente sur MSX1.

0016Eh (Main-ROM) **NSETRD** New SET address for ReaD

Rôle : Transfert dans le VDP de l'adresse de la case mémoire où doit s'effectuer la lecture.
Entrée : HL = Adresse de lecture (00000h~0FFFFh)
Sortie : Rien.
Modifie : AF.
Note : Routine absente sur MSX1.

00171h (Main-ROM) **NSTWRT** New SeT address for WRiTe

Rôle : Transfert dans le VDP de l'adresse de la case mémoire où doit s'effectuer l'écriture.
Entrée : HL = Adresse d'écriture (00000h~0FFFFh).
Sortie : Rien.
Modifie : AF.
Note : Routine inexistante sur MSX1.

00174h (Main-ROM) **NRDVRM** New ReaD VRaM

Rôle : Lecture d'une case mémoire de VRAM.
Entrée : HL = Adresse de la case mémoire à lire (00000h~0FFFFh).
Sortie : A = Contenu de la case mémoire lue.
Modifie : F.
Note : Routine inexistante sur MSX1.

00177h (Main-ROM) ***NWRVRM*** New WRite VRaM

- Rôle : Écriture dans une case mémoire de VRAM.
Entrée : HL = Adresse de la case mémoire (00000h~0FFFFh).
 A = Donnée à écrire.
Sortie : Rien.
Modifie : AF.
Note : Routine existante qu'à partir du MSX1.

Routines ajoutées aux MSX2+ :

0017Ah (Main-ROM) ***RDRES*** ReaD RESet

- Rôle : Indique le type initialisation du MSX paramétré.
Entrée : Rien.
Sortie : A = Le bit 7 à 1 indique si la routine STARUP (0000h) effectue un « Soft Reset » sinon ce sera un « Hard Reset ». Le bit 5 indique, sur MSX turbo R, quel CPU utiliser à l'initialisation (1 pour le R800).
Modifie : AF.
Note : - Routine existante qu'à partir du MSX2+.
 - Un « Hard Reset » initialise toute la RAM. Le logo du MSX s'affiche pour marquer la différence.

0017Dh (Main-ROM) ***WRRES*** WRite RESet

- Rôle : Choix du type initialisation du MSX.
Entrée : A = Le bit 7 permet de choisir la façon d'initialiser le MSX. 1 pour « Soft Reset » sinon « Hard Reset ». Sur MSX turbo R, mettre le bit 5 à 1 pour initialiser avec le R800.
Sortie : Rien.
Modifie : AF.
Note : - Routine existante qu'à partir du MSX2+.
 - Un « Hard Reset » initialise toute la RAM. Le logo du MSX s'affiche pour marquer la différence.

Routines ajoutées aux MSX turbo R :

00180h (Main-ROM) **CHGCPU** CHanGe CPU

Rôle : Changer le CPU du MSX turbo R. (Z80A à 3,579Mhz ou R800 à 7.159Mhz.)

Entrée : A = Paramètres.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
LED	0	0	0	0	0	MODE	

MODE = 00 pour Z80 ; 01 pour R800 en mode ROM et 10 pour R800 en mode DRAM.

LED = 1 allume la LED si changement vers R800 sinon, éteint la LED. 0 pour laisser tel quel.

Sortie : Rien.

Modifie : Rien.

Note : - Routine pour MSX turbo R seulement.

- En mode RAM, le contenu des ROM du système (Main-ROM, Sub-ROM et Kanji Driver ROM) est recopié en RAM. Cette zone de mémoire est protégée en écriture. Ce mode permet d'accélérer l'exécution des programmes mais il fait perdre les quatre dernières pages du Memory Mapper à l'utilisateur.

- Le changement de CPU peut s'effectuer à tout moment du Z80A au R800 en mode ROM et vice versa. Cependant, le passage en mode RAM doit s'effectuer au démarrage du MSX.

- L'utilisation du R800 sous MSX-DOS1 peut endommager les fichiers manipulés.

00183h (Main-ROM) **GETCPU** GET CPU

Rôle : Indique quel est le CPU en fonctionnement.

Entrée : Rien.

Sortie : A = 0 si Z80 ; 1 si R800 en mode ROM et 2 si R800 en mode DRAM.

Modifie : F.

Note : Routine pour MSX turbo R seulement.

00186h (Main-ROM) **PCMPLY** PCM PLaY

Rôle : Jouer un son numérisé.

Entrée : A = Paramètres.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
SOR	0	0	0	0	0	FREQ	

FREQ = Fréquence de l'échantillonnage.

00=15.75kHz ; 01=7.875kHz ; 10=5.25kHz et 11=3.9375kHz.

SOR = Emplacement des données à lire.

0 pour RAM principale ; 1 pour VRAM.

BC = Longueur des données à lire.

HL = Adresse du début des données à lire. (08000h~ si RAM principale)

E = 3 bits de point fort de l'adresse des données à lire. (VRAM seulement)

Sortie : A = 1 si la fréquence de l'échantillonnage est erronée ;

2 si la lecture a été stoppée en pressant la touche « STOP ».

F = Indicateur C à 1 si la lecture s'est déroulée normalement.

E = 3 bits de point fort de l'adresse où la lecture a été stoppée. (VRAM)

HL = Adresse où la lecture a été stoppée.

Modifie : Tout.

Notes : - Routine pour MSX turbo R seulement.

- Cette routine utilise toujours le R800. Si le Z80A est utilisé lors de l'appel à cette routine, le R800 sera utilisé en mode ROM puis replacera le Z80A avant de rendre la main.

- Les interruptions sont coupées pendant l'exécution de cette routine.

00189h (Main-ROM) **PCMREC** PCM RECOrd

Rôle : Numériser un son.

Entrée : A = Paramètres.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
DEST	Trigger level				COMP	FREQ	

FREQ = Fréquence d'échantillonnage.

00=15.75kHz, 01=7.875kHz, 10=5.25kHz et 11=3.9375kHz.

COMP = 1 pour compression des données.

Trigger level = Niveau de déclenchement (sensibilité).

DEST = Emplacement des données à écrire.

0 pour RAM principale. 1 pour VRAM.

BC = Longueur des données à écrire.

HL = Adresse du début des données à écrire. (08000H~ si RAM principale)

E = 3 bits de point fort de l'adresse des données à écrire. (VRAM seulement)

Sortie : A = 1 si la fréquence de l'échantillonnage est erronée.

2 si la lecture a été stoppée en pressant la touche « STOP ».

F = Indicateur C à 1 si la lecture s'est déroulée normalement.

E = 3 bits de point fort de l'adresse où la lecture a été stoppée. (VRAM)

HL = Adresse où la lecture a été stoppée.

Modifie : Tout.

Notes : - Routine pour MSX turbo R seulement.

- Cette routine utilise toujours le R800. Si le Z80A est utilisé lors de l'appel à cette routine, le R800 sera utilisé en mode ROM puis remplacera le Z80A avant de rendre la main.

- Les interruptions sont coupées pendant l'exécution de cette routine.

En plus des routines du Bios, quelques routines du Basic qui sont garanties par ASCII (« How to use graphic routines of MSX » du 05/02/86) :

058FCh (Main-ROM) **LINE** LINE

- Rôle : Tracé d'une ligne.
- Entrée : GXPOS (0FCB3h) = Abscisse du pixel de départ.
 GYPOS (0FCB5h) = Ordonnée du pixel de départ.
 BC = Abscisse du pixel d'arrivée.
 DE = Ordonnée du pixel d'arrivée.
- Sortie : Rien.
- Modifie : BC, DE, HL.
- Notes : - Il est nécessaire d'appeler SETATR (0011Ah) avant d'utiliser cette routine.
 - LINE fonctionne dans tous les modes graphiques.

058C1h (Main-ROM) **BOXFIL** BOX FIL1

- Rôle : Tracé d'un rectangle plein.
- Entrée : GXPOS (0FCB3h) = Abscisse du pixel de départ.
 GYPOS (0FCB5h) = Ordonnée du pixel de départ.
 BC = Abscisse du pixel d'arrivée.
 DE = Ordonnée du pixel d'arrivée.
- Sortie : Rien.
- Modifie : HL.
- Note : Il est nécessaire d'appeler SETATR (0011Ah) avant d'utiliser cette routine.
 BOXFIL fonctionne dans les SCREEN 2 à 4.

057F5h (Main-ROM) **PSET** Pixel SET

- Rôle : Placer un pixel à l'écran.
- Entrée : BC = Abscisse du pixel à placer.
 DE = Ordonnée du pixel à placer.
- Sortie : Rien.
- Modifie : HL.
- Notes : - Il est nécessaire d'appeler SETATR (0011Ah) avant d'utiliser cette routine.
 - PSET fonctionne dans tous les modes graphiques.

059E3h (Main-ROM) *PAINT* PAINT

- Rôle : Remplissage d'une surface à l'écran.
- Entrée : BC = Abscisse du pixel où commence le remplissage.
DE = Ordonnée du pixel où commence le remplissage.
A = Couleur de motif à remplir. (Idem couleur de peinture de SCREEN 2.)
- Sortie : Rien.
- Modifie : HL.
- Notes : - Il est nécessaire d'appeler SETATR (0011Ah) puis PNTINI (00129h) avant d'utiliser cette routine.
- PAINT fonctionne dans tous les modes graphiques.

Exemple d'utilisation :

```
10 CLEAR 100,&HD000
20 AD=&HD000
30 READ A$:IF a$="FIN" THEN 100
40 POKE AD,VAL("&H"+A$):AD=AD+1:GOTO 30
100 SCREEN 5:COLOR 2,1,1:CLS:LINE(&H5F,&H5F)-
(&H90,&H90),7,B:DEFUSR=&HD000:PRINT USR(0)
110 GOTO 110
500 DATA 3E,09,CD,1A,01,01,80,00,11,80,00
510 DATA 3E,07,CD,29,01,CD,E3,59,C9,FIN
```

Le programme en assembleur contenu les DATA :

```
PEINDRE: LD A,9 ; Couleur pour peindre
CALL SETATR
LD BC,080H
LD DE,080H
LD A,7 ; Couleur du contour
CALL PNTINI
CALL PAINT
RET
```

3.3 Table des sauts du Bios de la Sub-ROM

Voici le Bios de la Sub-ROM. Cette ROM contient un Bios qui permet manipuler les routines supplémentaires spécifiques au MSX2 ou aux générations supérieures. Pour appeler une de ces routines, vous utilisez la routine EXTROM (0015Fh) ou SUBROM (0015Ch) de la Main-ROM.

Liste des routines de la mémoire Morte auxiliaire (« Sub-ROM ») :

00069h (Sub-ROM) ***PAINT*** PAINT

- Rôle : Remplissage d'une surface à l'écran.
Entrée : HL = Pointeur sur le texte de l'interpréteur du Basic.
Sortie : HL = Pointeur réactualisé.
Modifie : Tout.
Notes : - Valide dans les SCREEN 5 à 8 / 12.
 - Utilisé par l'interpréteur du Basic.

Exemple d'utilisation :

```
                  ORG 0C000H
;
EXTROM           EQU 0015FH
PAINT            EQU 00069H
;
DEBUT:           LD HL, DATA
                  LD IX, PAINT
                  CALL EXTROM
                  RET
DATA:            DB '(100,100),5,8'
                  DB 00H
                  END DEBUT
```

0006Dh (Sub-ROM) ***PSET*** Pixel SET

- Rôle : Modifier un pixel à l'écran.
Entrée : HL = Pointeur sur le texte Basic.
Sortie : HL = Pointeur réactualisé.
Modifie : Tout.
Notes : - Valide dans les SCREEN 5 à 8 / 12.
 - Utilisé par l'interpréteur du Basic.
 - Voir l'exemple de PAINT ci-dessus pour le fonctionnement.

00085h (Sub-ROM) **DOGRPH** DO GRaPHic line

Rôle : Tracé de ligne.

Entrée : BC = Abscisse du pixel de départ.

DE = Ordonnée du pixel de départ.

GXPOS (0FCB3h) = Abscisse du pixel d'arrivée.

GYPOS (0FCB5h) = Ordonnée du pixel d'arrivée.

ATRBYT (0F3F2h) = Couleur du tracé (0 ~ 3 en SCREEN 6 ; 0 ~ 15 dans les
SCREEN 5 et 7 ; 0 ~ 255 dans les SCREEN 8 à 12).

LOGOPR (0FB02h) = Opérateur logique à appliquer.

Sortie : Rien.

Modifie : AF.

Note : Valide dans les SCREEN 5 à 8 / 12.

LOGOPR (« logical operation ») peut être :

0 (0000B) = IMP	8 (1000B) = TIMP
1 (0001B) = AND	9 (1001B) = TAND
2 (0010B) = OR	10 (1010B) = TOR
3 (0011B) = XOR	11 (1011B) = TXOR
4 (0100B) = NOT	12 (1100B) = TNOT

Les opérateurs de droite sont identiques à ceux de gauche sauf qu'ils n'agissent pas quand la couleur de fond est 0.

Exemple d'utilisation :

```

                                ORG 0C000H
;
EXTROM EQU 0015FH
DOGRPH EQU 00085H
GXPOS EQU 0FCB3H
GYPOS EQU 0FCB5H
ATRBYT EQU 0F3F2H
LOGOPR EQU 0FB02H
;
DEBUT: LD BC,10H
        LD DE,50H
        LD HL,90H
        LD (GXPOS),HL
        LD (GYPOS),HL
        LD A,7
        LD (ATRBYT),A
        XOR A
        LD (LOGOPR),A
        LD IX,DOGRPH
        CALL EXTROM
        RET
        END DEBUT
```


00091h (Sub-ROM) **MAPXYC** MAP X&Y to Current

- Rôle : Convertie les coordonnées graphiques contenues dans BC et DE en une adresse mémoire vidéo et un masque. (Pas utile dans les SCREEN 5 à 8).
- Entrée : BC = Abscisse.
 DE = Ordonnée.
- Sortie : HL et CLOC (0F92Ah) = En SCREEN 3, adresse correspondante aux coordonnées. Dans les SCREEN 5 à 8, abscisse.
 A et CMASK (0F92Ch) = En SCREEN 3, masque à appliquer à l'octet ci-dessus pour placer le pixel qui nous intéresse. Dans les SCREEN 5 à 8, ordonnée du pixel à placer.
- Modifie : F.
- Note : Valide dans les SCREEN 3 et 5 à 8.

00095h (Sub-ROM) **READC** READ Current

- Rôle : Lecture de l'attribut du pixel actuel.
- Entrée : CLOC (0F92Ah) = En SCREEN 3, adresse où se trouve le curseur graphique en VRAM. Dans les SCREEN 5 à 8, abscisse actuelle du curseur graphique.
 CMASK (0F92Ch) = En SCREEN 3, masque à appliquer. Dans les SCREEN 5 à 8, ordonnée actuelle du curseur graphique.
- Sortie : A = Attribut. (Numéro de la couleur du pixel.)
- Modifie : AF.
- Note : Valide dans les SCREEN 3 et 5 à 8.

00099h (Sub-ROM) **SETATR** SET ATRibute

- Rôle : Modification de l'octet d'attribut ATRBYT (0F3F2h).
- Entrée : A = Valeur du nouvel attribut. (Couleur de tracé des routines graphiques.)
- Sortie : F = Indicateur C à 1 si attribut non valide (>3 en SCREEN 6 ; >15 en SCREEN 5 et 7).
- Modifie : F.

0009Dh (Sub-ROM) **SETCSET** Current

- Rôle : Allumage du pixel actuel.
- Entrée : CLOC (0F92Ah) = En SCREEN 3, adresse où se trouve le curseur graphique en VRAM. Dans les SCREEN 5 à 8, abscisse actuelle du curseur graphique.
CMASK (0F92Ch) = En SCREEN 3, masque à appliquer. Dans les SCREEN 5 à 8, ordonnée actuelle du curseur graphique.
ATRBYT (0F3F2h) = Couleur du pixel à tracer (0 ~ 3 en SCREEN 6 ; 0 ~ 15 dans les SCREEN 3 à 5 et 7 ; 0 ~ 255 dans les SCREEN 8 à 12).
- Sortie : Rien.
- Modifie : AF.
- Note : Valide dans les SCREEN 3 et 5 à 8 / 12.

000A1h (Sub-ROM) **TRIGHTC** Test RIGHT Current

- Rôle : Déplacement d'un pixel vers la droite après vérification de la validité des coordonnées.
- Entrée : CLOC (0F92Ah) et CMASK (0F92Ch) = Coordonnées du pixel actuel.
- Sortie : CLOC (0F92Ah) et CMASK (0F92Ch) = Nouvelles coordonnées.
- Modifie : AF.
- Note : Valide en SCREEN 3 uniquement.

000A5h (Sub-ROM) **RIGHTC** RIGHT Current

- Rôle : Déplacement d'un pixel vers la droite
- Entrée : CLOC (0F92Ah) et CMASK (0F92Ch) = Coordonnées du pixel actuel.
- Sortie : CLOC (0F92Ah) et CMASK (0F92Ch) = Nouvelles coordonnées.
- Modifie : AF.
- Note : Valide en SCREEN 3 uniquement.

000A9h (Sub-ROM) **TLEFTC** Test LEFT Current

- Rôle : Déplacement d'un pixel vers la gauche après vérification de la validité des coordonnées.
- Entrée : CLOC (0F92Ah) et CMASK (0F92Ch) = Coordonnées du pixel actuel.
- Sortie : CLOC (0F92Ah) et CMASK (0F92Ch) = Nouvelles coordonnées.
- Modifie : AF.
- Note : Valide en SCREEN 3 et 5 à 8.

000ADh (Sub-ROM) **LEFTC** LEFT Current

Rôle : Déplacement d'un pixel vers la gauche
Entrée : CLOC (0F92Ah) et CMASK (0F92Ch) = Coordonnées du pixel actuel.
Sortie : CLOC (0F92Ah) et CMASK (0F92Ch) = Nouvelles coordonnées.
Modifie : AF
Note : Valide en SCREEN 3 uniquement.

000B1h (Sub-ROM) **TDOWNC** Test DOWN Current

Rôle : Déplacement d'un pixel vers le bas après vérification de la validité des coordonnées.
Entrée : CLOC (0F92Ah) et CMASK (0F92Ch) = Coordonnées du pixel actuel.
Sortie : CLOC (0F92Ah) et CMASK (0F92Ch) = Nouvelles coordonnées.
Modifie : AF.
Note : Valide en SCREEN 3 et 5 à 8.

000B5h (Sub-ROM) **DOWNC** DOWN Current

Rôle : Déplacement d'un pixel vers le bas.
Entrée : CLOC (0F92Ah) et CMASK (0F92Ch) = Coordonnées du pixel actuel.
Sortie : CLOC (0F92Ah) et CMASK (0F92Ch) = Nouvelles coordonnées.
Modifie : AF.
Note : Valide en SCREEN 3 uniquement.

000B9h (Sub-ROM) **TUPC** Test UP Current

Rôle : Déplacement d'un pixel vers le haut après vérification de la validité des coordonnées.
Entrée : CLOC (0F92Ah) et CMASK (0F92Ch) = Coordonnées du pixel actuel.
Sortie : CLOC (0F92Ah) et CMASK (0F92Ch) = Nouvelles coordonnées.
Modifie : AF.
Note : Valide en SCREEN 3 et 5 à 8.

000BDh (Sub-ROM) **UPC** UP Current

- Rôle : Déplacement d'un pixel vers le haut.
Entrée : CLOC (0F92Ah) et CMASK (0F92Ch) = Coordonnées du pixel actuel.
Sortie : CLOC (0F92Ah) et CMASK (0F92Ch) = Nouvelles coordonnées.
Modifie : AF.
Note : Valide en SCREEN 3 uniquement.

000C1h (Sub-ROM) **SCANR** SCAN Right

- Rôle : Recherche vers la droite du premier pixel d'une autre couleur.
Entrée : DE = Nombre de pixels sur lesquels la recherche doit être effectuée.
Sortie : DE = Position du pixel recherché.
Modifie : Tout.
Note : Voir SCANR en Main-ROM pour plus d'explications.

000C5h (Sub-ROM) **SCANL** SCAN Left

- Rôle : Recherche vers la gauche du premier pixel d'une autre couleur.
Entrée : DE = Nombre de pixels sur lesquels la recherche doit être effectuée.
Sortie : DE = Position du pixel recherché.
Modifie : Tout.
Note : Voir SCANL en Main-ROM pour plus d'explications.

000C9h (Sub-ROM) **NVXLN**

- Rôle : Tracé d'un rectangle à l'écran.
Entrée : BC = Abscisse du pixel de départ.
DE = Ordonnée du pixel de départ.
GXPOS (0FCB3h) = Abscisse du pixel d'arrivée.
GYPOS (0FCB5h) = Ordonnée du pixel d'arrivée.
ATRBYT (0F3F2h) = Couleur de tracé (0 ~ 3 en SCREEN 6 ; 0 ~ 15 en SCREEN 5 et 7 ; 0 ~ 255 dans les SCREEN 8 à 12).
LOGOPR (0FB02h) = Opérateur logique à appliquer.
Sortie : Rien.
Modifie : AF.
Notes : - Valide dans les SCREEN 5 à 8 / 12.
- Voir DOGRPH (00085h en Sub-ROM) pour l'explication de LOGOPR.

000CDh (Sub-ROM) ***NV BXFL***

- Rôle : Tracé d'un rectangle plein à l'écran.
- Entrée : BC = Abscisse du pixel de départ.
 DE = Ordonnée du pixel de départ.
 GXPOS (0FCB3h) = Abscisse du pixel d'arrivée.
 GYPOS (0FCB5h) = Ordonnée du pixel d'arrivée.
 ATRBYT (0F3F2h) = Couleur de tracé (0 ~ 3 en SCREEN 6 ; 0 ~ 15 en SCREEN
 5 et 7 ; 0 ~ 255 dans les SCREEN 8 à 12).
 LOGOPR (0FB02h) = Opérateur logique à appliquer.
- Sortie : Rien.
- Modifie : AF
- Notes : - Valide dans les SCREEN 5 à 8 / 12.
 - Voir DOGRPH (00085h en Sub-ROM) pour l'explication de LOGOPR

000D1h (Sub-ROM) ***CHG MOD*** CHanGe MODE

- Rôle : Changement de mode d'écran (SCREEN X).
- Entrée : A = SCREEN désiré. (0 ~ 8 / 12)
- Sortie : SCRMOD (0FCAFh) = Nouveau mode d'écran.
- Modifie : Tout.
- Note : La palette n'est pas initialisée par cette routine, utilisez la routine CHGMDP
(001B5h en Sub-ROM) si vous avez besoin de l'initialisation de la palette.

000D5h (Sub-ROM) ***INITXT*** INIitialize TeXT mode

- Rôle : Initialisation du mode texte 40x24 (SCREEN 0)
- Entrée : TXTNAM (0F3B3h) = Adresse de la table des caractères
 TXTCGP (0F3B7h) = Adresse de la table des formes
- Sortie : Rien.
- Modifie : Tout.
- Note : Il n'est pas obligatoire de définir TXTNAM et TXTCGP car le MSX initialise leur
 valeur à l'allumage.

000D9h (Sub-ROM) **INIT32** INIitialize Text 32 mode

Rôle : Initialisation du mode texte 32x24 (SCREEN 1)
Entrée : T32NAM (0F3BDh) = Adresse de la table des caractères
 T32COL (0F3BFh) = Adresse de la table des couleurs
 T32CGP (0F3C1h) = Adresse de la table des formes
 T32ATR (0F3C3h) = Adresse de la table des attributs de Sprites
 T32PAT (0F3C5h) = Adresse de la table des formes de Sprites
Sortie : Rien.
Modifie : Tout.
Note : Il n'est pas obligatoire de définit T32NAM, etc car le MSX initialise leur valeur à l'allumage.

000DDh (Sub-ROM) **INIGRP** INIitialize GRaPhic mode

Rôle : Initialisation du mode graphique 256x192 (SCREEN 2).
Entrée : GRPNAM (0F3C7h) = Adresse de la table des motifs.
 GRPCOL (0F3C9h) = Adresse de la table des couleurs.
 GRPCGP (0F3CBh) = Adresse de la table des formes.
 GRPATR (0F3CDh) = Adresse de la table des attributs de Sprites.
 GRPPAT (0F3CFh) = Adresse de la table des formes de Sprites.
Sortie : Rien.
Modifie : Tout.
Note : Il n'est pas obligatoire de définit GRPNAM, etc car le MSX initialise leur valeur à l'allumage.

000E1h (Sub-ROM) **INIMLT** INIitialize MuLTicolor mode

Rôle : Initialisation du mode SCREEN 3.
Entrée : MLTNAM (0F3D1h) = Adresse de la table des caractères.
 MLTCOL (0F3D3h) = Adresse de la table des couleurs.
 MLTCGP (0F3D5h) = Adresse de la table des formes.
 MLTATR (0F3D7h) = Adresse de la table des attributs de Sprites.
 MLTPAT (0F3D9h) = Adresse de la table des formes de Sprites.
Sortie : Rien.
Modifie : Tout.
Note : Il n'est pas obligatoire de définit MLTNAM, etc car le MSX initialise leur valeur à l'allumage.

000E5h (Sub-ROM) **SETTXT** SET TeXT mode

- Rôle : Passage direct en mode texte 40x24.
- Entrée : TXTNAM (0F3B3h) = Adresse de la table des caractères.
 TXTCGP (0F3B7h) = Adresse de la table des formes.
- Sortie : Rien.
- Modifie : Tout.
- Note : Il n'est pas obligatoire de définir TXTNAM et TXTCGP car le MSX initialise leur valeur à l'allumage.

000E9h (Sub-ROM) **SETT32** SET Text 32 mode

- Rôle : Passage direct en mode texte 32x24.
- Entrée : T32NAM (0F3BDh) = Adresse de la table des caractères.
 T32COL (0F3BFh) = Adresse de la table des couleurs.
 T32CGP (0F3C1h) = Adresse de la table des formes.
 T32ATR (0F3C3h) = Adresse de la table des attributs de Sprites.
 T32PAT (0F3C5h) = Adresse de la table des formes de Sprites.
- Sortie : Rien.
- Modifie : Tout.
- Note : Il n'est pas obligatoire de définir T32NAM, etc car le MSX initialise leur valeur à l'allumage.

000EDh (Sub-ROM) **SETGRP** SET GRaPhic mode

- Rôle : Passage direct en mode SCREEN 2.
- Entrée : GRPNAM (0F3C7h) = Adresse de la table des caractères.
 GRPCOL (0F3C9h) = Adresse de la table des couleurs.
 GRPCGP (0F3CBh) = Adresse de la table des formes.
 GRPATR (0F3CDh) = Adresse de la table des attributs de Sprites.
 GRPPAT (0F3CFh) = Adresse de la table des formes de Sprites.
- Sortie : Rien.
- Modifie : Tout.
- Note : Il n'est pas obligatoire de définir GRPNAM, etc car le MSX initialise leur valeur à l'allumage.

000F1h (Sub-ROM) **SETMLT** SET MuLTicolor mode

- Rôle : Passage direct en mode SCREEN 3.
- Entrée : MLTNAM (0F3D1h) = Adresse de la table des caractères.
MLTCOL (0F3D3h) = Adresse de la table des couleurs.
MLTCGP (0F3D5h) = Adresse de la table des formes.
MLTATR (0F3D7h) = Adresse de la table des attributs de Sprites.
MLTPAT (0F3D9h) = Adresse de la table des formes de Sprites.
- Sortie : Rien.
- Modifie : Tout.
- Note : Il n'est pas obligatoire de définir MLTNAM, etc car le MSX initialise leur valeur à l'allumage.

000F5h (Sub-ROM) **CLRSPR** CLear Sprites

- Rôle : Initialisation des Sprites.
- Entrée : SCRMOD (0FCAFh) = Mode d'écran actuel.
- Sortie : Rien.
- Modifie : Tout.
- Note : La table des formes de Sprites est effacée, les numéros de Sprites sont mis aux numéros de plans, la couleur des Sprites est mise à celle de la couleur définie par FORCLR (0F3E9h) et la position des Sprites est réglée à 217.

000F9h (Sub-ROM) **CALPAT** CALculate address of PATtern table

- Rôle : Calcul de l'adresse du début des formes d'un Sprite.
- Entrée : A = Numéro de plan du Sprite.
- Sortie : HL = Adresse cherchée.
- Modifie : AF, DE, HL.
- Note : Idem CALPAT en Main-ROM.

000FDh (Sub-ROM) **CALATR** CALculate address of ATRibute table

- Rôle : Calcul de l'adresse du début des attributs d'un Sprite.
- Entrée : A = Numéro de plan du Sprite
- Sortie : HL = Adresse cherchée.
- Modifie : AF, DE, HL.
- Note : Idem CALATR en Main-ROM.

00101h (Sub-ROM) **GSPSIZ** Get Sprite SIZE

Rôle : Indique la taille actuel des Sprites.
Entrée : Rien.
Sortie : A = Taille d'un Sprite. (8 ou 16).
 F = L'indicateur C est mis à 1 si la taille des Sprites est 16x16 ; 0 si 8x8.
Modifie : AF.
Note : Idem GSPSIZ en Main-ROM.

00105h (Sub-ROM) **GETPAT** GET PATtern

Rôle : Recherche du pattern d'un caractère.
Entrée : A = Code ASCII du caractère.
Sortie : PATWRK (0FC40h) = Les huit octets qui définissent le pattern du caractère.
Modifie : Tout.

00109h (Sub-ROM) **WRTVRM** WRiTe VRaM

Rôle : Écriture dans une case mémoire de la VRAM
Entrée : HL = Adresse de case mémoire. (0 ~ 0FFFFh)
 A = Donnée à écrire.
Sortie : Rien.
Modifie : AF.
Note : Cette routine permet l'accès à toute la VRAM, au contraire du WRTVRM de la Main-ROM.

0010Dh (Sub-ROM) **RDVRM** ReaD VRaM

Rôle : Lecture d'une case mémoire de la VRAM
Entrée : HL = Adresse de case mémoire (0 ~ 0FFFFh)
Sortie : A = Contenu de la case mémoire lue.
Modifie : AF.
Note : Cette routine permet l'accès à toute la VRAM, au contraire du RDVRM de la Main-ROM.

00111h (Sub-ROM) **CHGCLR** CHanGe CoLoR

Rôle : Réactualise les couleurs de l'écran avec les valeurs par défaut.
Entrée : FORCLR (0F3E9h) = Couleur de texte ou de tracé.
 BAKCLR (0F3EAh) = Couleur de fond.
 BDRCLR (0F3EBh) = Couleur de bordure.
Sortie : Rien.
Modifie : Tout.
Notes : - En mode graphique, seule la couleur de bordure change.
 - Même fonction que CHGCLR (00062h) de la Main-ROM.

00115h (Sub-ROM) **CLS** CLear Screen

Rôle : Effacement de l'écran.
Entrée : Rien.
Sortie : Rien.
Modifie : Tout.

00119h (Sub-ROM) **CLRTXT** CLear TeXT screen

Rôle : Effacement de l'écran en mode texte.
Entrée : Rien.
Sortie : Rien.
Modifie : Tout.

0011Dh (Sub-ROM) **DSPFNK** DiSPlay FuNction Keys

Rôle : Affichage des touches de fonction.
Entrée : Rien.
Sortie : Rien.
Modifie : Tout.
Note : Valide en SCREEN 0 et 1 uniquement.

00121h (Sub-ROM) **DELLN0** DELeTe LiNe mode 0

Rôle : Effacement d'une ligne en mode texte.
Entrée : L = Numéro de la ligne
Sortie : Rien.
Modifie : Tout.
Note : Valide en SCREEN 0 uniquement.

00125h (Sub-ROM) **INSLN0** INSert LiNe mode 0

Rôle : Insertion d'une ligne en mode texte.
Entrée : L = Numéro de la ligne.
Sortie : Rien.
Modifie : Tout.
Note : Valide en SCREEN 0 uniquement.

00129h (Sub-ROM) **PUTVRM** PUT character in VRaM

Rôle : Affichage d'un caractère en mode texte.
Entrée : C = Code ASCII du caractère à afficher.
 H = Numéro de colonne.
 L = Numéro de ligne.
Sortie : Rien.
Modifie : AF.

0012Dh (Sub-ROM) **WRTVDP** WRiTe VDP

Rôle : Écriture dans un registre du VDP.
Entrée : C = Numéro de registre (0 ~ 23 et 32 ~ 46).
 B = Donnée à écrire.
Sortie : Rien.
Modifie : AF, BC

00131h (Sub-ROM) **VDPSTA** VDP STATus

Rôle : Lecture d'un des registres de statut du VDP.
Entrée : A = Numéro de registre (0 ~ 9).
Sortie : A = Contenu du registre lu.
Modifie : F

00135h (Sub-ROM) **KYKLOK** KeY Kana LOcK

Rôle : Traitement du voyant et des touches KANA.
Entrée : Rien.
Sortie : Rien.
Modifie : AF
Note : Sans grand intérêt pour les MSX disponibles en France.

00139h (Sub-ROM) **PUTCHR** PUT kana CHaRacter

Rôle : Détection de l'appui sur une touche du clavier et transformation en code KANA.
Entrée : F = Indicateur Z à 1 si pas de conversion.
Sortie : Rien.
Modifie : Tout.
Note : Sans grand intérêt pour les MSX disponibles en France.

0013Dh (Sub-ROM) **SETPAG** SET PAGE

Rôle : Changement de page graphique.
Entrée : ACPAGE = Page affichée à l'écran.
 DPPAGE = Page sur laquelle on travaille.
Sortie : Rien.
Modifie : AF.

00141h (Sub-ROM) **INIPLT** INItialize PaLeTte

Rôle : Initialisation de la palette de couleurs.
Entrée : Rien.
Sortie : Rien.
Modifie : AF, BC, DE.
Note : Équivalent de l'instruction Basic « COLOR=NEW ».

00145h (Sub-ROM) **RSTPLT** ReSTore PaLeTte

Rôle : Récupération de la palette depuis la VRAM.
Entrée : Rien.
Sortie : Rien.
Modifie : AF, BC, DE.
Note : Équivalent de l'instruction Basic « COLOR=RESTORE ».

00149h (Sub-ROM) **GETPLT** GET PaLeTte

Rôle : Récupération des valeurs la palette d'une couleur.
Entrée : A = Numéro de la couleur.
Sortie : B = 4 bits de poids fort pour le rouge, 4 bits de poids faible pour le bleu.
 C = 4 bits de poids faible pour le vert.
Modifie : AF, DE.

0014Dh (Sub-ROM) **SETPLT** SET PaLeTte

Rôle : Modification d'une couleur dans la palette.
Entrée : D = Numéro de la couleur (0 ~ 15).
 A = 4 bits de poids fort pour le rouge, 4 bits de poids faible pour le bleu.
 E = 4 bits de poids faible pour le vert.
Sortie : Rien.
Modifie : AF.

00151h (Sub-ROM) **PUTSPR** PUT Sprite

Rôle : Affichage d'un Sprite.
Entrée : HL = Pointeur de texte du Basic.
Sortie : HL = Pointeur réactualisé.
Modifie : Tout.
Note : Utilisé par l'interpréteur du Basic.

00155h (Sub-ROM) **COLOR** COLOR

Rôle : Changement de couleurs, de la palette, de couleurs de Sprite.
Entrée : HL = Pointeur de texte du Basic.
Sortie : HL = Pointeur réactualisé.
Modifie : Tout.
Note : Utilisé par l'interpréteur du Basic.

00159h (Sub-ROM) **SCREEN** SCREEN

Rôle : Modification de tous les paramètres définis par l'instruction Basic. « SCREEN ».
Entrée : HL = Pointeur de texte du Basic.
Sortie : HL = Pointeur réactualisé.
Modifie : Tout.
Note : Utilisé par l'interpréteur du Basic.

0015Dh (Sub-ROM) **WIDTHS** WIDTH Screen

Rôle : Modification de la largeur de l'écran.
Entrée : HL = Pointeur de texte du Basic.
Sortie : HL = Pointeur réactualisé.
Modifie : Tout.
Note : Utilisé par l'interpréteur du Basic.

00161h (Sub-ROM) **VDP** VDP
Rôle : Écriture dans un registre du VDP.
Entrée : HL = Pointeur de texte du Basic.
Sortie : HL = Pointeur réactualisé.
Modifie : Tout.
Note : Utilisé par l'interpréteur du Basic.

00165h (Sub-ROM) **VDPF** VDP Fetch
Rôle : Lecture d'un registre du VDP.
Entrée : HL = Pointeur de texte du Basic
Sortie : HL = Pointeur réactualisé
Modifie : Tout.
Note : Utilisé par l'interpréteur du Basic.

00169h (Sub-ROM) **BASEBASE**
Rôle : Écriture dans un registre « base » du VDP.
Entrée : HL = Pointeur de texte du Basic.
Sortie : HL = Pointeur réactualisé.
Modifie : Tout.
Note : Utilisé par l'interpréteur du Basic.

0016Dh (Sub-ROM) **BASEF** BASE Fetch
Rôle : Lecture d'un registre « base » du VDP.
Entrée : HL = Pointeur de texte du Basic.
Sortie : HL = Pointeur réactualisé.
Modifie : Tout.
Note : Utilisé par l'interpréteur du Basic.

00171h (Sub-ROM) **VPOKE** Video POKE
Rôle : Écriture dans la mémoire vidéo.
Entrée : HL = Pointeur de texte du Basic.
Sortie : HL = Pointeur réactualisé.
Modifie : Tout.
Note : Utilisé par l'interpréteur du Basic.

00175h (Sub-ROM) **VPEEK** Video PEEK

Rôle : Lecture de la mémoire vidéo.
Entrée : HL = Pointeur de texte du Basic.
Sortie : HL = Pointeur réactualisé.
Modifie : Tout.
Note : Utilisé par l'interpréteur du Basic.

00179h (Sub-ROM) **SETS** SET Screen

Rôle : Traitement des instructions « SET » (SCREEN, ADJUST, TIME, etc).
Entrée : HL = Pointeur de texte du Basic.
Sortie : HL = Pointeur réactualisé.
Modifie : Tout.
Note : Utilisé par l'interpréteur du Basic.

0017Dh (Sub-ROM) **BEEP** BEEP

Rôle : Émission d'un bref bip sonore.
Entrée : Rien.
Sortie : Rien.
Modifie : Tout.

00181h (Sub-ROM) **PROMPT** PROMPT

Rôle : Affichage de OK ou du message défini par l'utilisateur.
Entrée : Rien.
Sortie : Rien.
Modifie : Tout.

00185h (Sub-ROM) **SDFSCR**

Rôle : Récupération des options (largeur d'écran, couleurs, etc) sauvegardés dans la RAM CMOS de l'horloge.
Entrée : Rien.
Sortie : Rien.
Modifie : Tout.

00189h (Sub-ROM) **SETSCR** SET SCReen

Rôle : Idem SDFSCR avec en plus l'affichage du message de départ à l'allumage.

Entrée : Rien.

Sortie : Rien.

Modifie : Tout.

0018Dh (Sub-ROM) **SCOPY** Screen COPY

Rôle : Traitement de l'instruction « COPY » vidéo.

Entrée : HL = Pointeur de texte du Basic.

Sortie : HL = Pointeur réactualisé

Modifie : Tout.

00191h (Sub-ROM) **BLTVV** BLock Transfer Vram to Vram

Rôle : Transfert d'un bloc d'un endroit de mémoire vidéo à un autre.

Entrée : HL = 0F562h (pointeur).

Sortie : Rien.

Modifie : Tout.

Exemple d'utilisation :

Faire « COPY (32,16)-(128,96) TO (192,128) » en assembleur.

```

                                ORG 0C000H
EXTROM            EQU 0015FH
BLTVV            EQU 00191H
SX               EQU 0F562H ; abscisse pixel de départ
SY               EQU SX+2    ; ordonnée pixel de départ
DX               EQU SY+2    ; abscisse pixel de destination
DY               EQU DX+2    ; ordonnée pixel de destination
NX               EQU DY+2    ; largeur du bloc à copier
NY               EQU NX+2    ; hauteur du bloc à copier
ARG              EQU NY+3
LOGOP            EQU ARG+1   ; opérateur logique
;
DEBUT:           LD HL,20H
                 LD (SX),HL
                 LD HL,10H
                 LD (SY),HL
                 LD HL,0C0H
                 LD (DX),HL
                 LD HL,80H
                 LD (DY),HL
                 LD HL,80H-20H+1 ; calcul de la largeur
                 LD (NX),HL
                 LD HL,60H-10H+1 ; calcul de la hauteur
                 LD (NY),HL
                 XOR A            ; ARG doit toujours être
                 LD (ARG),A      ; mis à zéro !
                 LD A,3
                 LD (LOGOP),A    ; au même format que LOGOPR
;
; ne pas oublier ceci
;
                 LD HL,SX
                 LD IX,BLTVV
                 CALL EXTROM
                 RET
                 END DEBUT
```

Le VDP travaille comme s'il n'y avait qu'un écran géant dans lequel on déplace une fenêtre suivant le schéma :

SCREEN 5		VRAM	SCREEN 6	
(0,0)	(255,0)	00000h	(0,0)	(511,0)
page 0			page 0	
(0,255)	(255,255)	07FFFh	(0,255)	(511,255)
(0,256)	(255,256)	08000h	(0,256)	(511,256)
page 1			page 1	
(0,511)	(255,511)	0FFFFh	(0,511)	(511,511)
(0,512)	(255,512)	10000h	(0,512)	(511,512)
page 2			page 2	
(0,767)	(255,767)	17FFFh	(0,767)	(511,767)
(0,768)	(255,768)	18000h	(0,768)	(511,768)
page 3			page 3	
(0,1023)	(255,1023)	1FFFFh	(0,1023)	(511,1023)

SCREEN 7		VRAM	SCREEN 8 ~ 12	
(0,0)	(511,0)	00000h	(0,0)	(255,0)
page 0			page 0	
(0,255)	(511,255)	0FFFFh	(0,255)	(255,255)
(0,256)	(511,256)	10000h	(0,256)	(255,256)
page 1			page 1	
(0,511)	(511,511)	1FFFFh	(0,511)	(255,511)

00195h (Sub-ROM) **BLTVM** BLock Transfert Vram from Memory

Rôle : Transfert de la mémoire centrale vers la mémoire vidéo

Entrée : HL = 0F562h (pointeur).

Sortie : Rien.

Modifie : Tout.

Exemple d'utilisation :

Faire « COPY &HD000 TO (16,32) » en assembleur.

```

                                ORG 0C000H
EXTROM            EQU 0015FH
BLTVM            EQU 00195H
DPTR            EQU 0F562H ; adresse en mémoire centrale
DX              EQU DPTR+4 ; abscisse pixel de destination
DY              EQU DX+2  ; ordonnée pixel de destination
ARG             EQU DY+7
LOGOP           EQU ARG+1 ; opérateur logique
;
DEBUT:           LD HL,0D000H
                 LD (DPTR),HL
                 LD HL,10H
                 LD (DX),HL
                 LD HL,20H
                 LD (DY),HL
                 XOR A                    ; ARG doit toujours être
                 LD (ARG),A              ; mis à zéro !!!
                 LD A,2
                 LD (LOGOP),A            ; au même format que LOGOPR
;
; ne pas oublier ceci
;
                 LD HL,DPTR
                 LD IX,BLTVM
                 CALL EXTROM
                 RET
;
                 ORG 0D000H
P.  HORZ        DW 030H                  ; nombre de pixels horizontaux
P.  VERT        DW 00BH                  ; nombre de pixels verticaux
NBOCT          DS 210H                  ; nombre d'octets nécessaires
END DEBUT
```

Pour calculer NBOCT :

en SCREEN 5 - (NBOCT) = (P. HORZ)/2 * (P.VERT) +1

en SCREEN 6 - (NBOCT) = (P. HORZ)/4 * (P.VERT) +1

en SCREEN 7 - (NBOCT) = (P. HORZ)/2 * (P.VERT) +1

en SCREEN 8 - (NBOCT) = (P. HORZ) * (P.VERT)

Pour plus de précisions, voir la routine BLTVV.

00199h (Sub-ROM) *BLTMV* BLock Transfert Memory from Vram

Rôle : Transfert de la mémoire vidéo vers la mémoire centrale.

Entrée : HL = 0F562h (pointeur).

Sortie : Rien.

Modifie : Tout.

Exemple d'utilisation :

Faire « COPY (16,32) TO &HD000 » en assembleur.

```

                ORG 0C000H
EXTROM         EQU 0015FH
BLTMV         EQU 00199H
SX            EQU 0F562H ; abscisse du pixel de départ
SY            EQU SX+2   ; ordonnée du pixel de départ
DPTR         EQU SY+2   ; adresse en mémoire centrale
NX            EQU SY+6   ; abscisse du pixel de destination
NY            EQU NX+2   ; ordonnée du pixel de destination
ARG          EQU NY+3
LOGOP        EQU ARG+1 ; opérateur logique
;
DEBUT:        LD HL,0D000H
                LD (DPTR),HL
                LD HL,10H
                LD (SX),HL
                LD HL,20H
                LD (SY),HL
                LD HL,60H-10H+1 ; calcul de la largeur
                LD (NX),HL
                LD HL,40H-20H+1 ; calcul de la hauteur
                LD (NY),HL
                XOR A ; ARG doit toujours être
                LD (ARG),A ; mis à zéro !!!
                LD A,2
                LD (LOGOP),A ; au même format que LOGOPR
;
; ne pas oublier ceci
;
                LD HL,SX
                LD IX,BLTMV
                CALL EXTROM
                RET
                END DEBUT
```

Vous trouverez en 0D000h et 0D001h la largeur du bloc, en 0D002h et 0D003h la hauteur du bloc. Pour calculer le nombre d'octets utilisés à partir de 0D004h pour stocker le bloc, appliquez la formule adéquate :

en SCREEN 5 - nb octets = largeur/2 * hauteur +1

en SCREEN 6 - nb octets = largeur/4 * hauteur +1

en SCREEN 7 - nb octets = largeur/2 * hauteur +1

en SCREEN 8 - nb octets = largeur * hauteur

Pour plus de précisions, voir la routine BLTVV.

0019Dh (Sub-ROM) **BLTVD** BLock Transfert Vram from Disk

Rôle : Transfert de la disquette vers la mémoire vidéo.

Entrée : HL = 0F562h (pointeur).

Sortie : Rien.

Modifie : Tout.

Exemple d'utilisation :

Faire « COPY "A:ESSAI.SC7" to (16,32) » en assembleur.

```

                                ORG 0C000H
EXTROM            EQU 0015FH
BLTVD            EQU 0019DH
FNPTR            EQU 0F562H        ; pointeur du nom de fichier
DX               EQU FNPTR+4        ; abscisse de destination
DY               EQU DX+2           ; ordonnée de destination
ARG              EQU DY+7
LOGOP            EQU ARG+1         ; opérateur logique
;
DEBUT:            LD HL,NOM
                   LD (FNPTR),HL
                   LD HL,10H
                   LD (DX),HL
                   LD HL,20H
                   LD (DY),HL
                   XOR A             ; mis à zéro !!!
                   LD (ARG),A
                   LD (LOGOP),A     ; au même format que LOGOPR
;
; ne pas oublier ce qui suit.
;
                   LD HL,FNPTR
                   LD IX,BLTVD
                   CALL EXTROM
                   RET
;
NOM:              DB 022H,'A:ESSAI.SC7',022H,000H
                   END DEBUT
```

Si vous appelez cette routine depuis le Basic, faites-le par un
DEFUSR=&HC000 : A\$=USR(0). N'oubliez pas le dollar. Dans A\$,
vous aurez le nom du fichier.

Pour plus de précisions, voir la routine BLTVV.

001A1h (Sub-ROM) *BLTDV* BLock Transfert Disk from Vram

Rôle : Transfert de la mémoire vidéo vers la disquette.

Entrée : HL = 0F562h (pointeur).

Sortie : Rien.

Modifie : Tout.

Exemple d'utilisation :

Faire « COPY (16,32)-(96,128) TO "A:ESSAI.SC7" » en assembleur.

```

                                ORG 0C000H
EXTROM            EQU 0015FH
BLTDV            EQU 001A1H
SX               EQU 0F562H ; abscisse du pixel de départ
SY               EQU SX+2    ; ordonnée du pixel de départ
FNPTR            EQU SY+2    ; pointeur sur le nom de fichier
NX               EQU SY+6    ; abscisse du pixel de destination
NY               EQU NX+2    ; ordonnée du pixel de destination
ARG              EQU NY+3
;
DEBUT:            LD HL,NOM
                   LD (FNPTR),HL
                   LD HL,10H
                   LD (SX),HL
                   LD HL,20H
                   LD (SY),HL
                   LD HL,60H-10H+1 ; calcul de la largeur
                   LD (NX),HL
                   LD HL,80H-20H+1 ; calcul de la hauteur
                   LD (NY),HL
                   XOR A            ; ARG doit toujours être
                   LD (ARG), A      ; mis à zéro !!!
;
; ne pas oublier ceci
;
                   LD HL, SX
                   LD IX, BLTDV
                   CALL EXTROM
                   RET
;
NOM:              DB 022H, 'A:ESSAI.SC7', 022H, 000H
                   END DEBUT
```

Si vous appelez cette routine depuis le Basic, faites-le par un
DEFUSR=&HC000 : A\$=USR(0). N'oubliez pas le dollar. Dans A\$,
vous aurez le nom du fichier.

Pour plus de précisions, voir la routine BLTVV.

001A5h (Sub-ROM) **BLTMD** BLock Transfert Memory from Disk

Rôle : Transfert de la disquette vers la mémoire centrale.

Entrée : HL = 0F562h (pointeur).

Sortie : Rien.

Modifie : Tout.

Exemple d'utilisation :

Faire « COPY "A:ESSAI.SC7" TO <0D000h> » en assembleur.

```

                                ORG 0C000H
EXTROM            EQU 0015FH
BLTMD            EQU 001A5H
FNPTR            EQU 0F562H            ; pointeur du nom de fichier
SPTR            EQU FNPTR+4            ; adresse de départ en mémoire
EPTR            EQU FNPTR+6            ; adresse de fin en mémoire
;
DEBUT:           LD HL,NOM
                 LD (FNPTR),HL
                 LD HL,0D000H
                 LD (SPTR),HL
                 LD HL,0D020H
                 LD (EPTR),HL
;
; ne pas oublier ceci
;
                 LD HL,FNPTR
                 LD IX,BLTMD
                 CALL EXTROM
                 RET
;
NOM:            DB 022H,'A:ESSAI.SC7',022H,000H
                END DEBUT
```

Si vous appelez cette routine depuis le Basic, faites-le par un
DEFUSR=&HC000 : A\$=USR(0). N'oubliez pas le dollar. Dans A\$,
vous aurez le nom du fichier.

001A9h (Sub-ROM) **BLTDM** BLock Transfert Disk from Memory

Rôle : Transfert de la mémoire centrale vers la disquette.

Entrée : HL = 0F562h (pointeur).

Sortie : Rien.

Modifie : Tout.

Exemple d'utilisation :

Faire « COPY <0D000h> TO "A:ESSAI.SC7" » en assembleur.

```

                                ORG 0C000H
EXTROM            EQU 0015FH
BLTDM            EQU 001A9H
SPTR             EQU 0F562H ; adresse de départ en mémoire
EPTR             EQU SPTR+2 ; adresse de fin en mémoire
FNPTR            EQU SPTR+4 ; pointeur sur le nom de fichier
;
DEBUT:            LD HL,NOM
                  LD (FNPTR),HL
                  LD HL,0D000H
                  LD (SPTR),HL
                  LD HL,0D020H
                  LD (EPTR),HL
;
; ne pas oublier ceci
;
                  LD HL,SPTR
                  LD IX,BLTDM
                  CALL EXTROM
                  RET
;
NOM:              DB 022H,'A:ESSAI.SC7',022H,000H
                  END DEBUT
```

Si vous appelez cette routine depuis le Basic, faites-le par un
DEFUSR=&HC000 : A\$=USR(0). N'oubliez pas le dollar. Dans A\$,
vous aurez le nom du fichier.

001ADh (Sub-ROM) **NEWPAD** NEW get touch PAD

- Rôle : Lecture de la tablette graphique, du stylo optique, de la souris ou du Track-ball (Cat).
- Entrée : A = Numéro de l'opération à effectuer (0 ~ 19).
- Sortie : A = résultat de l'opération.
- Modifie : Tout.
- Notes : - Sur MSX2, il est possible d'appeler indifféremment cette routine ou GTPAD en Main-ROM. NEWPAD est un peu plus rapide.
- Voici la liste des opérations possibles :

Entrée	Port	Valeur résultante
0	Général n°1	0FFh si le stylet est en contact avec la tablette graphique, sinon 0.
1		Abscisse de la tablette
2		Ordonnée de la tablette
3		0FFh si le bouton du stylet est pressé, sinon 0.
4	Général n°2	0FFh si le stylet est en contact avec la tablette graphique, sinon 0.
5		Abscisse de la tablette
6		Ordonnée de la tablette
7		0FFh si le bouton du stylet est pressé, sinon 0.
8	Crayon optique	0FFh si le crayon optique touche l'écran, sinon 0.
9		Abscisse du crayon optique.
10		Ordonnée du crayon optique.
11		0FFh si le bouton du crayon optique est pressé, sinon 0.
12	Général n°1	Toujours à 0FFh.
13		Abscisse de la souris ou du Track-Ball.
14		Ordonnée de la souris ou du Track-Ball.
15		Toujours à 0. (inutilisé)
16	Général n°2	Toujours à 0FFh.
17		Abscisse de la souris ou du Track-Ball.
18		Ordonnée de la souris ou du Track-Ball.
19		Toujours à 0. (inutilisé)

- Les valeurs données par le crayon optique ne sont valables que lorsque le crayon optique touche l'écran.
- Effectuer les opérations 12 ou 16 juste avant d'effectuer une lecture des coordonnées d'une souris ou d'un Track-Ball sinon, les valeurs obtenues n'auront aucun sens.
- Pour tester l'état des boutons de la souris ou du Track-Ball, veuillez utiliser la routine GTTRIG (000DBh en Main-ROM).

- Le MSX turbo R ne gère pas le crayon optique. (A = 0 en sortie.)

001B1h (Sub-ROM) **GETPUT** GET time/date & PUT kanji

Rôle : Récupération de l'heure et de la date et traitement des caractères kanji.
Entrée : HL = Pointeur de texte du Basic.
Sortie : Rien.
Modifie : Tout.

001B5h (Sub-ROM) **CHGMDP** CHanGe Mode DisPlay

Rôle : Changement de mode d'écran (SCREEN X).
Entrée : A = Numéro du mode d'écran (0 ~ 8 / 12).
Sortie : Rien.
Modifie : Tout.
Note : Par rapport à CHGMOD (000D1h), cette routine initialise la palette.

001B9h (Sub-ROM) **RESV1** RESerVed 1

Rôle : Réserve aux versions futures de MSX.
Entrée : Rien.
Sortie : Rien.
Modifie : Rien.

001BDh (Sub-ROM) **KNJPRT** KaNji PRinT

Rôle : Affichage d'un caractère kanji à l'écran.
Entrée : BC = Code du caractère kanji.
 A = Type d'affichage.
Sortie : Rien.
Modifie : AF.
Note : Routine présente sur les MSX japonais ayant une Kanji-ROM.

001F5h (Sub-ROM) **REDCLK** REaD CLoCK

Rôle : Lecture de la mémoire vive non-volatile (S-RAM) de l'horloge interne.
Entrée : C = Adresse mémoire.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
-	-	Mode		Adresse (0 ~ 12)			

Sortie : A = Contenu de la case mémoire lue. (quatre bits de poids faible)
Modifie : F.

Note : L'adresse mémoire se décompose ainsi :

En mode 0 :

Adresse	bit 3	bit 2	bit 1	bit 0
00	Compteur de l'unité des secondes			
01	-	Compteur de la dizaine des secondes		
02	Compteur de l'unité des minutes			
03	-	Compteur de la dizaine des minutes		
04	Compteur de l'unité des heures			
05	-	-	Compteur de la dizaine des heures	
06	Compteur du jour des semaines			
07	Compteur de l'unité des jours des mois			
08	-	-	Compteur de la dizaine des jours des mois	
09	Compteur de l'unité des mois			
10	-	-	-	Dizaine des mois
11	Compteur de l'unité des années			
12	Compteur de la dizaine des années			

En mode 2 :

Adresse	bit 3	bit 2	bit 1	bit 0
00	Identificateur (1010 si la RAM est initialisée)			
01	Décalage horizontale du SET ADJUST (-8 à 7)			
02	Décalage verticale du SET ADJUST (-8 à 7)			
03	-	-	-	SCREEN 0 / 1
04	4 bits de poids faible de la valeur de WIDTH			
05	-	3 bits de poids fort de la valeur de WIDTH		
06	Couleur du texte par défaut (0 ~ 15)			
07	Couleur de fond par défaut (0 ~ 15)			
08	Couleur de bordure par défaut (0 ~ 15)			
09	Vitesse de transfert 0 = 1200 baud 1 = 2400 baud	0 = Imprimante MSX 1 = Imprimante Non MSX	Clic des touches	KEY OFF / ON
10	Type de BEEP		Volume du BEEP	
11	-	-	Couleur de la page de présentation	
12	Code de la zone : 0 = Japon ; 1 = USA ; 2 = International ; 3 = Grande Bretagne ; 4 = France ; 5 = Allemagne ; 6 = Italie ; 7 = Espagne ; 8 = Emirats arabes ; 9 = Corée ; 10 = URSS ; 11 ~ 15 = Indéfinis (au 05/02/1986)			

Note : Le code de la zone n'est pas toujours le bon par défaut.

En mode 3 :

Adresse	bit 3	bit 2	bit 1	bit 0
00	identificateur (0 ~ 15) : 0 = TITLE ; 1 = PASSWORD ; 2 = PROMPT ; 3 à 15 = <i>Indéfinis</i> au 05/02/1986			
01	Réservé pour l'utilisateur			
02	Réservé pour l'utilisateur			
03	Réservé pour l'utilisateur			
04	Réservé pour l'utilisateur			
05	Réservé pour l'utilisateur			
06	Réservé pour l'utilisateur			
07	Réservé pour l'utilisateur			
08	Réservé pour l'utilisateur			
09	Réservé pour l'utilisateur			
10	Réservé pour l'utilisateur			
11	Réservé pour l'utilisateur			
12	Réservé pour l'utilisateur			

Exemple d'utilisation :

```
ORG 0C000H
;
EXTROM EQU 0015FH
REDCLK EQU 001F5H
;
DEBUT: LD C,02CH ; mode 2 - adresse 12
LD IX,REDCLK
CALL EXTROM
RET ; lors du retour A contient 4
```

001F9h (Sub-ROM) **WRCLK** WRiTe CLock

Rôle : Écriture dans la mémoire vive non-volatile de l'horloge interne.

Entrée : C = Adresse mémoire

A = Donnée à écrire. (quatre bits de poids faible)

Sortie : Rien.

Modifie : F.

Note : Voir REDCLK (ci-dessus) pour le détail du fonctionnement de cette routine.

3.4 Table des sauts du Bios de la Disk-ROM

La Disk-ROM a aussi une table des sauts. Cette ROM est présente dans tous les MSX ayant un lecteur de disquette interne ou dans les interfaces de disque. Elle contient toutes les routines d'accès aux disques du Disk Basic, le noyau du MSX-DOS 1 et le pilote du contrôleur de disque.

Mémoire Morte du contrôleur de disques : (« Disk-ROM »)

04010h (Disk-ROM) **DSKIO** DiSK I/O

- Rôle : Lire ou écrire de 1 à 255 secteur sur un disque.
- Entrée : A = Numéro de disque physique de l'interface correspondante à la Disk-ROM.
F = Mettre l'indicateur Carry à 0 pour lire ou 1 pour écrire.
B = Nombre de secteur à lire / écrire.
C = Identifiant du type de Média.
DE = Numéro du premier secteur.
HL = Adresse de destination en RAM.
- Sortie : A = Code d'erreur. (si il y a)
F = L'indicateur Carry passe à 1 pour indiquer qu'une erreur s'est produite.
B = Nombre de secteur à lu / écrit.
- Modifie : Tout.
- Note : Voici les codes d'erreur possible.
- 0 = Le disque est protégé contre l'écriture. (Write protected)
 - 2 = Le disque n'est pas inséré ou pas prêt. (Disk offline)
 - 4 = Erreur détectée à la vérification. (Data/CRC error)
 - 6 = Erreurs de recherche d'une piste sur le disque. (Seek error)
 - 8 = Secteur d'amorçage ne contient pas d'entête. Le disque nécessite un formatage. (Header not found)
 - 10 = Anomalie d'écriture. (Write fault / Write current error)
 - 12 = Erreur non identifiée.

Exemple en assembleur :

```
DAC      equ    0f7f6h
DRVINV   equ    0fb21h
DSKIO    equ    04010h
ENASLT   equ    00024h
EXPTBL   equ    0fcc1h
VALTYP   equ    0f663h

; --- Entête du fichier

                db    0feh                ; Fichier binaire code
```

```

        dw    START      ; Adresse de destination du programme
        dw    END        ; Adresse de fin du programme
        dw    START      ; Adresse d'execution du programme
; ---
        ORG    0c000h

START:   ld     a,2        ; Lecteur C:
        ld     hl,DRVINV  ; Table des interfaces de disque
RPT:    sub    (hl)
        jr     c,CONT
        inc   hl
        inc   hl
        jr     RPT
CONT:   add    a,(hl)
        push  af
        inc   hl
        ld    a,(hl)
        ld    h,040h
        call  ENASLT      ; Selectionne la Disk-Rom du disque C:
        ei
        pop   af
READ:   ld     b,7
        ld     c,0fbh    ;
        ld     de,5
        ld     hl,0c100h
        or    a          ; Met carry à 0 (pour lecture)
        call  DSKIO
        ld    hl,255
        jr    nc,RD_OK
        ld    l,a
RD_OK:  ld     (DAC+2),hl ; Code d'erreur -> variable du Basic
        ld     a,2
        ld     (VALTYP),a
        ld     a,(EXPTBL)
        ld     h,040h
        call  ENASLT      ; Sélectionne la Main-Rom
        ei
        ret           ; Retour au Basic
END:

```

Une fois assemblé et sauvegardé sous le nom « RDSECT5.BIN », exécuter la routine avec le programme Basic suivant.

```

10 CLEAR200,&HBFFF:DEFUSR=&HC000
20 BLOAD"RDSECT5.BIN"
30 A%=USR(0): IF A%<255 THEN
40 FOR I=&HC100 TO I+7*512-1 STEP 32
50 FOR J=0 TO 10: C=PEEK(I+J)
60 IF C=0 THEN END ELSE IF C=&HE5 THEN 80
70 PRINT CHR$(C): NEXT J: PRINT
80 NEXT I: END
90 BEEP:ON A/2 GOTO 100,110,120,130,140
100 PRINT"DISK OFF LINE!":END
110 PRINT"DISK READ ERROR!":END
120 PRINT"DISK SEEK ERROR!":END
130 PRINT"SECTOR NOT FOUND!":END
140 PRINT"UNDEFINED ERROR!":END

```

04013h (Disk-ROM) ***DSKCHG*** DiSK CHanGe

Rôle : Vérifie l'état d'un changement de disque.

Entrée : A = Numéro de disque physique de l'interface correspondante à la Disk-ROM.

 B = Identifiant du type de Média.

 C = Identifiant du type de Média.

 HL = Adresse du FCB.

Sortie : A = Code d'erreur (voir DSKIO).

 F = L'indicateur Carry passe à 1 pour indiquer qu'une erreur s'est produite.

 B = 1 pour disque inchangé, 255 si il a été changé autrement, 0.

Modifie : Tout.

Note : Si le disque a été changé ou modifié (B=0), lire le secteur d'amorçage ou le secteur de la FAT pour un descripteur des médias de disque et transférer un nouveau DPB comme avec GETDPB.

04016h (Disk-ROM) **GETDPB** GET DPB

Rôle : Obtenir le bloc de paramètres d'un disque (Drive Parameter Block).

Entrée : A = Numéro de disque physique de l'interface correspondante à la Disk-ROM.
 B = Premier secteur de la FAT.
 C = Identifiant du type de Média.
 HL = Adresse du DPB à obtenir.

Sortie : HL = Adresse du DPB obtenu. (Longueur = 37 octets)

Modifie : Tout.

Description du DPB obtenu :

HL	Long.	Nom	
+0	1	MEDIA	Type de Média (F8h, ..., FFh)
+1	2	SECSIZ	Taille des secteurs
+3	1	DIRMSK	(SECSIZE/32)-1
+4	1	DIRSHFT	Nombre de bits dans DIRMSK
+5	1	CLUSMSK	(Secteurs par cluster)-1
+6	1	CLUSSHFT	(Nombre de bits dans CLUSMSK)+1
+7	2	FIRFAT	Numéro du secteur de la première FAT
+8	1	FATCNT	Nombre de FAT
+10	1	MAXENT	Nombre de fichier / dossier possible (254 max.)
+11	2	FIRREC	Numéro du premier secteur des données.
+13	2	MAXCLUS	
+15	1	FATSIZ	Nombre de secteur par FAT
+16	2	FIRDIR	

04019h (Disk-ROM) **CHOICE** CHOICE

Rôle : .

Entrée : HL = .

Sortie : HL = .

Modifie : Tout.

Notes :

0401Ch (Disk-ROM) **DSKFMT** DiSK ForMaT

Rôle : Formater un disque.

Entrée : HL = .

Sortie : HL = .

Modifie : Tout.

Note : Plusieurs interfaces utilisent un logiciel externe pour formater ses disques.

0401Fh (Disk-ROM)

Rôle : Stopper le moteur des disques de l'interface correspondante.
Entrée : Rien.
Sortie : Rien.
Modifie : Tout.
Note : Cette routine n'existe que si l'interface gère les disques amovibles. Dans le cas contraire, 0401Fh contiendra l'octet zéro (00h).

04029h (Disk-ROM) **MTOFF** MoTor OFF

Rôle : Stopper le moteur de tous les disques du système.
Entrée : Rien.
Sortie : Rien.
Modifie : Tout.
Note : Cette routine n'existe que si l'interface gère les disques amovibles. Dans le cas contraire, 04029h contiendra l'octet zéro (00h).
Exemple : Routine en assembleur.

```
rdslt      equ 000ch
calslt     equ 001ch
mtoff      equ 4029h
master_slot equ 0f348h

DiskOFF:
          ld  a, (master_slot)
          ld  hl, mtoff
          call rdslt
          and  a                ; Routine MTOFF présente?
          ret  z                ; Retour si pas de MTOFF
          ld  iy, (master_slot-1)
          ld  ix, mtoff
          jp  calslt
```

0402Dh (Disk-ROM) **GTSLT1** GET SLOt bank 1

Rôle : Donne le numéro de Slot sélectionné sur la plage 04000h~07FFFh.
Entrée : Rien.
Sortie : A = Numéro de Slot.
Modifie : Tout.

3.5 *Le Bios des extensions MSX*

La routine EXTBIO a été ajoutée au standard MSX afin de connaître combien et quelles extensions sont connectées au système. Elle permet aussi de les gérer plus aisément.

L'adresse de cette routine se trouve à 0FFCAh dans le Slot des variables système. La plupart des appels à EXTBIO provoquera un appel au Bios de l'extension correspondante.

Cette routine étant optionnelle, il est nécessaire de vérifier sa présence avant de l'appeler en lisant la variable HOKVLD (0FB20h). Si le bit 0 de cette variable est à zéro, c'est qu'il n'y a pas d'extension compatible.

0FFCAh ***EXTBIO***

Entrée : D = Identifiant de l'extension.

 E = Numéro de la fonction de la routine à appeler.

Note : De manière générale, la fonction 0 a pour but de créer une table pour connaître combien et quelles extensions sont connectées au système.

Il est possible d'ajouter une routine pour une extension en procédant de la façon suivante.

1. Lire le bit 0 de l'octet à l'adresse 0FB20h (HOKVLD). Si il est à 0, mettez-le à 1 puis, écrivez 0C9h (instruction RET) sur les 29 octets à partir de 0FFCAh.
2. Copier les 5 octets à partir de 0FFCAh vers une zone de votre choix.
3. Ensuite écrire les instructions de saut vers la routine de votre Bios à 0FFCAh. Votre routine devra se terminer par un saut vers la zone copiée auparavant (en 2).

Liste des périphériques avec leur fonctions :

000h (0) ***Broad-cast***

Fonction 0

Rôle : Création d'une table contenant les numéros de périphériques ayant un Bios étendus.

Entrée : B = Slot dans lequel créer la table.

 HL = Adresse de la table à créer.

Sortie : B = Slot dans lequel se trouve le premier octet suivant la table.

 HL = Adresse du premier octet suivant la table.

Modifie : Tout.

Note : La table résultante contient les numéros des périphériques disponibles chacun espacé d'un octet réservé (00h). Broad-cast (0) et System (255) n'apparaissent pas dans la table.

Voici un exemple pour obtenir la table des périphériques :

```
RAMAD0            equ    0f341h       ; Slot de la Main-Ram page 0~3fffh
```

```

extbio      equ      0ffcah    ; Adresse d'appel à un BIOS d'extension

getdev:
    ld      hl,table
    call   getslt    ; B = numéro le slot de la table
    ld      d,0      ; Broad-cast device
    ld      e,0      ; Fonction 'get device number'
    jp     extbio

; Routine getslt
; Entrée : HL = Adresse en Ram.
; Sortie : B = Numéro de Slot correspondant à l'adresse HL.
; Modifie : A et B.

getslt:
    push   hl
    ld     a,h
    rla
    rla
    rla    ; Bit 6 et 7 vers bit 1 et 0
    and   3    ; mise à zéro des bits inutilisées
    ld    c,a
    ld    b,0
    ld    hl,RAMAD0
    add   hl,bc
    ld    b,(hl) ; B = Numéro du Slot de la Main-RAM
    pop   hl
    ret

table:
    ds    32    ; Réserve 32 octets pour la table

```

Fonction 1

Rôle : Obtenir le numéro de trap à placer sur les instructions ON ... GOSUB du Basic afin d'insérer une routine personnelle en langage machine qui sera exécutée lors d'un saut effectué par un ON ... GOSUB.

Entrée : A = 0

Sortie : A = Numéro de trap

0 ~ 9 pour ON KEY GOSUB.

10 pour ON STOP GOSUB.

11 pour ON SPRITE GOSUB.

12 ~ 16 pour ON STRIG GOSUB.

17 pour ON INTERVAL GOSUB.

18 ~ 23 pour les périphériques étendus.

24 ~ 25 Réservés pour le système.

Modifie : F et DE.

Fonction 2

Rôle : Désactiver les interruptions des périphériques.

Entrée : Rien.

Sortie : Rien.

Note : Cette routine permet d'éviter les erreurs de transfert des périphériques qui envoient

un signal d'interruption. Cela arrive, par exemple, avec la RS-232C lorsqu'on coupe les interruptions du CPU trop longtemps.

Fonction 3

Rôle : Activer les interruptions des périphériques.
 Entrée : Rien.
 Sortie : Rien.

004h (4) *Memory Mapper*

Fonction 0

Rôle : Obtenir la table d'information sur le Memory Mapper primaire.
 Entrée : B = Slot dans lequel créer la table.
 HL = Adresse de la table à créer.
 Sortie : B = Slot dans lequel se trouve le premier octet suivant la table.
 HL = Adresse du premier octet suivant la table.
 Modifie : Tout.
 Note : La table d'information sur le Memory Mapper primaire a le format suivant.

Adresse entrée	
HL	Numéro du Slot du Memory Mapper primaire
HL+01h	8 bits de poids faible de l'adresse de la table des sauts
HL+02h	8 bits de poids fort de l'adresse de la table des sauts
HL+03h	Nombre de pages libres
HL+04h	Nombre de pages
HL+05h	Octet réservé
HL+06h	Octet réservé
HL+07h	Octet réservé
Adresse dans HL en sortie :	

Fonction 1

Rôle : Obtenir la table des variables des Memory Mapper.
 Entrée : A = 0
 Sortie : A = Numéro de Slot du Memory Mapper principal
 HL = Adresse de la table de variables
 Modifie : Tout.
 Note : Exemple de table de variables avec deux Memory Mapper connectés.

Adresse en sortie	
HL	Numéro du Slot du Memory Mapper
HL+01h	Nombre de pages
HL+02h	Nombre de pages libres
HL+03h	Nombre de pages réservées par le système
HL+04h	Nombre de pages réservées par l'utilisateur

HL+05h	Octet réservé
HL+06h	Octet réservé
HL+07h	Octet réservé
.	Numéro du Slot du second Memory Mapper
.	Nombre de pages
.	Nombre de pages libres
	Nombre de pages réservées par le système
	Nombre de pages réservées par l'utilisateur
	Octet réservé
	Octet réservé
	Octet réservé
	00h

Sur MSX turbo R, le Memory Mapper primaire sera toujours le Memory Mapper interne.

Fonction 2

Rôle : Obtenir la table des sauts vers les routines du Memory Mapper.

Entrée : A = 0

Sortie : A = Nombre de page du Memory Mapper primaire

B = Numéro de Slot du Memory Mapper primaire

C = Nombre de pages libres du Memory Mapper primaire

HL = Adresse de la table des sauts

Modifie : Tout sauf B.

Table des sauts :

HL + 00h ***ALL_SEG***

Rôle : Allouer une page.

Entrée : A = 0 pour l'utilisateur / 1 pour le système

B = Slot du mapper sous la forme FxxxSSPP.

Si xxx = 000, alloue une page au Slot indiqué.

Si xxx = 001, alloue une page dans un autre Slot que celui indiqué.

Si xxx = 010, alloue une page au Slot indiqué ou un autre si le mapper est déjà pris.

Si xxx = 011, alloue une page dans un autre Slot que celui indiqué. En cas d'échec, essaie dans un autre Slot.

B = 0 si Memory Mapper principal.

Sortie : F = Carry à 1 si aucune page libre.

Si Carry à 0, alors A = Numéro de page, B = Numéro de Slot du Memory Mapper (B = 0 si Memory Mapper principal.)

HL + 03h ***FRE_SEG***

Rôle : Libérer une page.

Entrée : A = Numéro de page.

B = Slot du mapper sous la forme FxxxSSPP (B = 0 pour le Memory Mapper principal.)

Sortie : F = Carry à 1 si aucune n'a été libérée.

- HL + 06h **RD_SEG**
Rôle : Lire un octet dans une page.
Entrée : A = Numéro de page.
 HL = Adresse à lire dans la page.
Sortie : A = Octet lu.
Modifie : AF.
Note : Cette routine désactive les interruptions.
- HL + 09h **WR_SEG**
Rôle : Ecrire un octet dans une page.
Entrée : A = Numéro de page.
 HL = Adresse.
 E = Octet à écrire.
Modifie : AF.
Note : Cette routine désactive les interruptions.
- HL + 0Ch **CAL_SEG**
Rôle : Appel inter-page.
Entrée : IYh = Numéro de page.
 IX = Adresse.
Modifie : Tout sauf AF, BC, DE et HL.
Note : AF, BC, DE et HL peuvent servir de paramètres pour la routine à appeler.
- HL + 0Fh **CALLS**
Rôle : Appel inter-page.
Entrée : Les trois octets suivants l'instruction call CALLS.
 call CALLS
 db PAGE
 dw ADRESSE
Modifie : Tout sauf AF, BC, DE et HL.
Note : AF, BC, DE et HL peuvent servir de paramètres pour la routine à appeler.
- HL + 12h **PUT_PH**
Rôle : Sélectionner une page sur la plage mémoire correspondante à l'adresse indiquée.
Entrée : HL = Adresse (ce sont en fait les bits 7 et 6 qui indiquent la plage 0~3FFFh, 4000h~7FFFh, 8000h~BFFFh ou C000h~FFFFh).
 A = Numéro de page.
Modifie : Rien.
- HL + 15h **GET_PH**
Rôle : Obtenir le numéro de page sélectionnée sur la plage mémoire correspondante à l'adresse indiquée.
Entrée : HL = Adresse (ce sont en fait les bits 7 et 6 qui indiquent la plage 0~3FFFh, 4000h~7FFFh, 8000h~BFFFh ou C000h~FFFFh).
 A = Numéro de page.
Modifie : Rien.
- HL + 18h **PUT_P0**
Rôle : Sélectionner une page sur la plage 0000h~3FFFh.
Entrée : A = Numéro de page.
- HL + 1bh **GET_P0**
Rôle : Obtenir le numéro de page de la plage 0000h~3FFFh.
Sortie : A = Numéro de page.

HL + 1eh **PUT_P1**

Rôle : Sélectionner une page sur la plage 4000h~7FFFh.
Entrée : A = Numéro de page.

HL + 21h **GET_P1**

Rôle : Obtenir le numéro de page de la plage 4000h~7FFFh.
Sortie : A = Numéro de page.

HL + 24h **PUT_P2**

Rôle : Sélectionner une page sur la plage 8000h~BFFFh.
Entrée : A = Numéro de page.

HL + 27h **GET_P2**

Rôle : Obtenir le numéro de page de la plage 8000h~BFFFh.
Sortie : A = Numéro de page.

HL + 2ah **PUT_P3**

Rôle : Sélectionner une page sur la plage C000h~FFFFh.
Entrée : A = Numéro de page.

HL + 2dh **GET_P3**

Rôle : Obtenir le numéro de page de la plage C000h~FFFFh.
Sortie : A = Numéro de page.

008h (8) **MSX-Modem and RS-232C**

Fonction 0

Rôle : Obtenir une table d'informations sur les MSX-Modem et RS-232C installés.

Entrée : B = Slot dans lequel créer la table.
HL = Adresse de la table à créer.

Sortie : B = Slot dans lequel se trouve le premier octet suivant la table.
HL = Adresse du premier octet suivant la table.

Modifie : Tout.

Note : Voici un exemple de table avec deux interfaces installées.

Adresse indiquée par HL en entrée :

Numéro du Slot de l'interface
Octet de poids faible de l'adresse des sauts
Octet de poids fort de l'adresse des sauts
Octet réservé (00h par défaut)
Numéro du Slot de la seconde interface
Octet de poids faible de l'adresse des sauts
Octet de poids fort de l'adresse des sauts
Octet réservé (00h par défaut)

Adresse indiquée par HL en sortie :

00Ah (10) **MSX-Audio**

Fonction 0

Rôle : Obtenir une table d'informations sur les MSX-Audio installés.
 Entrée : B = Slot dans lequel créer la table.
 HL = Adresse de la table à créer.
 Sortie : B = Slot dans lequel se trouve le premier octet suivant la table.
 HL = Adresse du premier octet suivant la table.
 Modifie : F.
 Note : Voici un exemple de table avec deux MSX-Audio installées.

Adresse indiquée par HL en entrée :	Numéro du Slot du MSX-Audio
	Octet de poids faible de l'adresse des sauts
	Octet de poids fort de l'adresse des sauts
	Numéro du Slot du second MSX-Audio
	Octet de poids faible de l'adresse des sauts
	Octet de poids fort de l'adresse des sauts
	Octet réservé (00h par défaut)
	Adresse indiquée par HL en sortie :

Fonction 1

Rôle : Obtenir la table des sauts vers les routines du MSX-Audio.
 Entrée : A = 0.
 B = Slot dans lequel créer la table.
 HL = Adresse de la table à créer.
 Sortie : A = Nombre de MSX-Audio installés
 HL = Adresse de la table des sauts.
 Modifie : Tout sauf B.

Table des sauts :

HL+00h **VERSION**

Rôle : Ces 3 octets servent à indiquer la version de la ROM.

HL+03h **MBIOS**

Rôle : Appel aux routines du MBIOS.

Entrée : HL = Adresse de la routine du MBIOS à appeler.

Pour les routines qui utilise IX et IY comme paramètre, veuillez placer ces paramètres dans BUF (0F55Eh) comme suit.

BUF = 8 bits de poids faible de IX.

BUF+1 = 8 bits de poids fort de IX.

BUF+2 = 8 bits de poids faible de IY.

BUF+3 = 8 bits de poids fort de IY.

Sortie : Voir les routine MBios.

HL+06h **AUDIO**

Rôle : Initialiser le MSX-Audio. Les routines du MSX-Audio ne seront utilisable qu'après une initialisation.

Entrée : Définir les paramètres suivants dans BUF (0F55Eh).

BUF = Nombre de chaine MML par voix FM pour PLAY. (0~9)

BUF+1 = Mode switch.

bit 0 pour activer la boîte à rythme

bit 1 pour activer l'ADPCM du PLAY

bit 2 pour activer le mode CMS du MSX-Audio

BUF+2 = Nombre de voix FM pour les instruments. (0~9)

BUF+3 = Nombre de voix FM de la première chaîne MML. (0~9)

BUF+4 = Nombre de voix FM de la seconde chaîne MML. (0~8)

BUF+5 = Nombre de voix FM de la troisième chaîne MML. (0~7)

BUF+6 = Nombre de voix FM de la quatrième chaîne MML. (0~6)

BUF+7 = Nombre de voix FM de la cinquième chaîne MML. (0~5)

BUF+8 = Nombre de voix FM de la sixième chaîne MML. (0~4)

BUF+9 = Nombre de voix FM de la septième chaîne MML. (0~3)

BUF+10 = Nombre de voix FM de la huitième chaîne MML. (0~2)

BUF+11 = Nombre de voix FM de la neuvième chaîne MML. (0~1)

Sortie : F = Carry passe à 1 en cas d'erreur de paramètre.

Note : Lorsque la boîte à rythme est activé le nombre de voix FM passe de 9 à 6 maximum.

HL+09h **SYNTH**

Rôle : Transfère le contrôle au programme d'application fourni. Cette routine n'aura aucun effet lorsque la routine AUDIO est appelé.

Entrée : Rien.

Sortie : Rien.

Note : Cette routine ne fonctionnera plus correctement après l'exécution de l'instruction CLEAR du Basic. (Il y a risque de plantage)

HL+0Ch **PLAYF**

Rôle : Examine de l'état de la lecture des MML.

Entrée : A = Numéro de voix. (0 pour toutes les voix)

Sortie : HL = FFFFh si lecture en cours sinon HL = 0000h.

HL+0Fh **BGM**

Rôle : Activer / désactiver le mode BGM (lecture en tâche de fond).

Entrée : A = Activer / désactiver le mode BGM. (1 / 0)

Sortie : Rien.

Note : Voici les fonctions qui peuvent fonctionner en mode BGM

- Jouer une chaîne MML.

- Lecture d'un enregistrement ADPCM du mode local.

- Lecture d'un enregistrement du clavier musical à l'adresse indiquée.

HL+12h **MKTEMPO**

Rôle : Réglage du tempo pour jouer ou enregistrer un air du clavier musical.

Entrée : DE = Nombre de noires par minute. (25 ~ 360)

Sortie : F = Carry passe à 1 en cas d'erreur de paramètre.

HL+15h **PLAYMK**

Rôle : Enregistrer un air au clavier musical.

Entrée : BC = Adresse de début des données à enregistrer au clavier musical.

DE = Adresse de fin des données à enregistrer au clavier musical.

Sortie : Rien.

HL+18h **RECMK**

Rôle : Enregistrer un air au clavier musical.

Entrée : BC = Adresse de début des données à enregistrer au clavier musical.

DE = Adresse de fin des données à enregistrer au clavier musical.

Sortie : Rien.

HL+1bh **STOPM**

Rôle : Stopper la lecture d'un enregistrement au clavier musical, de d'un enregistrement ADPCM / PCM, ou d'une chaine MML jouée.

Entrée : Rien.

Sortie : Rien.

Note : Il est possible de reprendre la lecture d'un enregistrement au clavier musical en appelant CONTMK.

HL+1eh **CONTMK**

Rôle : Continuer la lecture d'un enregistrement au clavier musical, ADPCM / PCM, ou d'une chaine MML jouée.

Entrée : Rien.

Sortie : Rien.

HL+21h **RECMOD**

Rôle : Régler le mode du clavier musical.

Entrée : A = Mode.

0 pour muet (N'enregistre pas).

1 pour enregistrement.

2 pour jouer l'air.

3 pour jouer tout en enregistrant.

Sortie : F = Carry passe à 1 en cas d'erreur de paramètre.

HL+24h **STPLY**

Rôle : Stopper la lecture d'une chaine MML.

Entrée : Rien.

Sortie : Rien.

HL+27h **SETPCM**

Rôle : Initialiser le fichier audio ADPCM/PCM.

Entrée : Définir les paramètres suivants dans BUF (0F55Eh).

BUF = Numéro de l'enregistrement. (0 ~ 15)

BUF+1 = Numéro de matériel à employer. (0 ~ 3 ou 5)

BUF+2 = Mode. (0 ou 1)

BUF+3 = Si le numéro de matériel est 1 ou 3 alors mettre ici un numéro de son en ROM. Si le numéro de matériel est 5, indiquer ici les 8 bits de poids faible de l'adresse en VRAM.

BUF+4 = Si le numéro de matériel est 1 ou 3 alors mettre 0. Si c'est 5, indiquer les 8 bits de poids fort de l'adresse en VRAM.

BUF+5 = 8 bits de poids faible de la longueur de l'enregistrement.

BUF+6 = 8 bits de poids fort de la longueur de l'enregistrement.

BUF+7 = 8 bits de poids faible de la fréquence d'échantillonnage.

BUF+8 = 8 bits de poids fort de la fréquence d'échantillonnage.

BUF+9 = Canal. (0 ou 1)

Sortie : F = Carry passe à 1 en cas d'erreur de paramètre.

Note : Le matériel 4 (le CPU) n'est pas utilisable.

HL+2ah **RECPCM**

Rôle : Enregistrer un son dans un fichier.

Entrée : Définir les paramètres suivants dans BUF (0F55Eh).

BUF = Numéro de l'enregistrement. (0~15)

BUF+1 = Synchro. (0 ou 1)

BUF+2 = Les 8 bits de faible fort du décalage.

BUF+3 = Les 8 bits de poids fort du décalage.

BUF+4 = 8 bits de poids faible de la longueur de l'enregistrement.

BUF+5 = 8 bits de poids fort de la longueur de l'enregistrement.

BUF+6 = 8 bits de poids faible de la fréquence d'échantillonnage.

BUF+7 = 8 bits de poids fort de la fréquence d'échantillonnage.

BUF+8 = Canal. (0 ou 1)

Sortie : F = Carry passe à 1 en cas d'erreur de paramètre.

Note : La fréquence d'échantillonnage peut être définie avec SETPCM. Si c'est le cas, mettre 0FFh à BUF+6 et BUF+7.

HL+2dh **PLAYPCM**

Rôle : Lire un enregistrement.

Entrée : Définir les paramètres suivants dans BUF (0F55Eh).

BUF = Numéro de l'enregistrement. (0 ~ 15)

BUF+1 = Lecture en boucle. (0 ou 1)

BUF+2 = 8 bits de faible du décalage.

BUF+3 = 8 bits de poids fort du décalage.

BUF+4 = 8 bits de poids faible de la longueur de l'enregistrement.

BUF+5 = 8 bits de poids fort de la longueur de l'enregistrement.

BUF+6 = 8 bits de poids faible de la fréquence d'échantillonnage.

BUF+7 = 8 bits de poids fort de la fréquence d'échantillonnage.

BUF+8 = Canal. (0 ou 1)

Sortie : F = Carry passe à 1 en cas d'erreur de paramètre.

Note : La fréquence d'échantillonnage peut être définie avec SETPCM. Si c'est le cas, mettre 0FFh à BUF+6 et BUF+7.

HL+30h **PCMFREQ**

Rôle : Changer la fréquence de lecture.

Entrée : BC = Fréquence du premier canal en Hz. (1800~49716)

DE = Fréquence du second canal en Hz. (1800~49716)

Sortie : F = Carry passe à 1 en cas d'erreur de paramètre.

HL+33h **MKPCM**

Rôle : Paramétrer le numéro de fichier ADPCM joué par le clavier musical.

Entrée : A = Numéro de l'enregistrement. (0~15 ou 0FFh pour aucun)

Sortie : F = Carry passe à 1 en cas d'erreur de paramètre.

Note : Seul les fichiers en mode local sont jouable.

HL + 36h **PCMVOL**

Rôle : Réglage du volume pour jouer un son ADPCM / PCM.

Entrée : BC = Volume du premier canal. (0~63)

DE = Volume du second canal. (0~63)

Sortie : F = Carry passe à 1 en cas d'erreur de paramètre.

Note : Lorsqu'il n'y a pas de deuxième canal, le registre DE doit contenir la même valeur que celle de BC. Par défaut, le volume ADPCM est 63, celui PCM est 32.

HL+39h **SAVEPCM**

Rôle : Sauvegarder un enregistrement ADPCM / PCM sur disque.

Entrée : A = Numéro de l'enregistrement.

HL = Adresse pointant sur le nom de fichier entre guillemets.

Sortie : F = Carry passe à 1 en cas de mauvais de numéro d'enregistrement.

Notes :

- En assembleur, on définit le nom de fichier de la façon suivante.
FILENAME: DB 22H, "A:TRACK.PCM", 22H, 0
- En cas d'erreur disque (nom de fichier erroné, disque non inséré, etc), la gestion sera transférée à la routine de traitement d'erreur de l'interpréteur BASIC. Vous pouvez l'intercepter grâce au Hook H.ERRO (0FFB1h).
- Le fichier créé sera constituer de la façon suivante.
En premier, il y a l'entête qui comprends 7 octets comme pour les fichiers binaires sauvegardés par l'instruction BSAVE du Basic suivi de 8 octets comprenant des informations sur les données ADPCM / PCM enregistrées.
Le reste sera les données du son numérisé.

Entête :

Octet du
fichier

Entête au format d'un fichier binaire

0	Indicateur de fichier binaire. (0FEh)
1~2	0000h
3~4	Longueur des données du fichier -1. (Bloc d'informations compris)
5~6	0000h pour ADPCM, 0001h pour PCM.

Bloc d'informations

7~8	Longueur des données de l'enregistrement. (Unité = 256 octets)
9~10	Fréquence d'échantillonnage en Herz.
11~12	8000h pour ADPCM (valeur par défaut), 0000h pour PCM.
13~14	007Fh pour ADPCM (valeur par défaut), 0000h pour PCM.

Bloc contenant les données du son numérisé

15~	Données du son numérisé
-----	-------------------------

HL+3Ch **LOADPCM**

Rôle : Chargement d'un enregistrement ADPCM / PCM sur disque effectué avec SAVEPCM.

Entrée : A = Numéro de l'enregistrement.

HL = Adresse pointant sur le nom de fichier entre guillemets.

Sortie : Rien.

Notes :

- En assembleur, on définit le nom de fichier de la façon suivante.
FILENAME: DB 22H, "A:TRACK.PCM", 22H, 0
- En cas d'erreur disque (nom de fichier erroné, disque non inséré, etc), la gestion sera transférée à la routine de traitement d'erreur de l'interpréteur BASIC. Vous pouvez l'intercepter grâce au Hook H.ERRO (0FFB1h).

HL+3Fh **COPYPCM**

Rôle : Copie les données d'un enregistrement ADPCM / PCM.

Entrée : Définir les paramètres suivants dans BUF (0F55Eh).

BUF = Numéro de l'enregistrement source. (0~15)

BUF+1 = Numéro de l'enregistrement de destination. (0~15)

BUF+2 = 8 bits de faible de l'adresse de l'enregistrement source.

BUF+3 = 8 bits de fort de l'adresse de l'enregistrement source.

BUF+4 = 8 bits de poids faible de la longueur de l'enregistrement.

BUF+5 = 8 bits de poids fort de la longueur de l'enregistrement.

BUF+6 = 8 bits de faible de l'adresse de l'enregistrement de destination.

BUF+7 = 8 bits de faible de l'adresse de l'enregistrement de destination.

BUF+8 = Canal source. (0 ou 1)

Sortie : F = Carry passe à 1 en cas d'erreur de paramètre.

HL+42h **CONVP**

Rôle : Convertir un enregistrement PCM au format ADPCM.

Entrée : Définir les paramètres suivants dans BUF (0F55Eh).

BUF = Numéro de l'enregistrement source. (0~15)

BUF+1 = Numéro de l'enregistrement de destination. (0~15)

Sortie : F = Carry passe à 1 en cas d'erreur de paramètre.

HL+45h **CONVA**

Rôle : Convertir un enregistrement ADPCM au format PCM.

Entrée : Définir les paramètres suivants dans BUF (0F55Eh).

BUF = Numéro de l'enregistrement source. (0~15)

BUF+1 = Numéro de l'enregistrement de destination. (0~15)

Sortie : F = Carry passe à 1 en cas d'erreur de paramètre.

HL+48h **VOICE**

Rôle : Définir la tonalité pour chaque voix sonore FM.

Entrée : Définir les paramètres suivants dans BUF (0F55Eh).

BUF = Bloc de paramètres de la Voix 1.

BUF+4 = Bloc de paramètres de la Voix 2.

BUF+8 = Bloc de paramètres de la Voix 3.

BUF+12 = Bloc de paramètres de la Voix 4.

BUF+16 = Bloc de paramètres de la Voix 5.

BUF+20 = Bloc de paramètres de la Voix 6.

BUF+24 = Bloc de paramètres de la Voix 7.

BUF+28 = Bloc de paramètres de la Voix 8.

BUF+32 = Bloc de paramètres de la Voix 9.

BUF+36 = Fin (0FFh)

Chaque bloc est constitué de 4 octets de la façon suivante.

+0 = Numéro de voix sonore FM. (0~8)

+1 = Mettre à 00h.

+2 = Numéro du son de la librairie. (0~63)

+3 = Mettre à 00h.

Certains blocs peuvent être constitués de la façon suivante.

+0 = Numéro de voix sonore FM. (0~8)

+1 = Mettre à FFh.

+2 = 8 bits de poids faible de l'adresse d'un enregistrement.

+3 = 8 bits de poids fort de l'adresse d'un enregistrement.

Sortie : Rien.

HL+4bh **VOICECOPY**

Rôle : Copier les données d'une voix FM.

Entrée : Définir les paramètres dans BUF (0F55Eh) selon une des cinq possibilités suivantes.

1. Copie un son de la librairie vers un autre son de la librairie.

BUF+0 = 0

BUF+1 = Numéro de son de la librairie source. (0~63)

BUF+2 = 0

BUF+3 = 0

BUF+4 = 0

BUF+5 = 0

BUF+6 = Numéro du son de la librairie de destination. (32~63)

BUF+7 = 0

BUF+8 = 0

BUF+9 = 0

2. Copie un son de la librairie vers un son de l'utilisateur.

BUF+0 = 0

BUF+1 = Numéro de son de la librairie source. (0~63)

BUF+2 = 0

BUF+3 = 0

BUF+4 = 0

BUF+5 = 0FFh

BUF+6 = 8 bits de poids faible de l'adresse de destination.

BUF+7 = 8 bits de poids fort de l'adresse de destination.

BUF+8 = 0

BUF+9 = 0

3. Copie un enregistrement vers un son de la librairie.

BUF+0 = 0FFh

BUF+1 = 8 bits de poids fort de l'adresse de l'enregistrement source.

BUF+2 = 8 bits de poids fort de l'adresse de l'enregistrement source.
BUF+3 = 0
BUF+4 = 0
BUF+5 = 0
BUF+6 = Numéro du son de la librairie de destination. (32~63)
BUF+7 = 0
BUF+8 = 0
BUF+9 = 0

4. Copie les sons de 32 à 63 de la librairie vers un enregistrement.

BUF = 0
BUF+1 = 0FFh
BUF+2 = 0
BUF+3 = 0
BUF+4 = 0
BUF+5 = 0FFh
BUF+6 = 8 bits de poids faible de l'adresse de destination.
BUF+7 = 8 bits de poids fort de l'adresse de destination.
BUF+8 = 8 bits de poids faible de la longueur de l'enregistrement.
BUF+9 = 8 bits de poids fort de la longueur de l'enregistrement.

5. Copie un enregistrement vers les sons de 32 à 63 de la librairie.

BUF = 0FFh
BUF+1 = 8 bits de poids faible de l'adresse de l'enregistrement source.
BUF+2 = 8 bits de poids fort de l'adresse de l'enregistrement source.
BUF+3 = 8 bits de poids faible de la longueur de l'enregistrement.
BUF+4 = 8 bits de poids fort de la longueur de l'enregistrement.
BUF+5 = 0
BUF+6 = 0FFh
BUF+7 = 0
BUF+8 = 0
BUF+9 = 0

Sortie : Rien.

010h (16)

MSX-JE

Fonction 0

Rôle : Définir la zone de travail d'entrée de caractères japonais. (Disponible sur certains MSX japonais ou, cartouches de Kanji et traitements de texte japonais avec MSX-JE.)

Entrée : B = Numéro du Slot de la zone de travail des caractères entrés par RETURN.
HL = Pointeur sur la zone de travail de 64 octets.

Sortie : B = Numéro du Slot suivant la zone de travail.
HL = Adresse suivant la zone de travail.

Modifie : F.

Notes : - La pile doit être sur la plage 0C000h~0FFFFh.

-

011h (17)

Kanji driver

Fonction 0

Rôle : ?

Entrée : ?
 Sortie : ?
 Modifie : ?
 Note : ?

0FFh (255) *System*

Fonction 0

Rôle : Obtenir une table d'informations sur les Bios étendu ajoutées au système.

Entrée : A = 0.
 B = Slot dans lequel créer la table.
 HL = Adresse de la table à créer.

Sortie : B = Slot dans lequel se trouve le premier octet suivant la table.
 HL = Adresse du premier octet suivant la table.

Modifie : Tout.

Note : Format de la table d'information sur les Bios étendu.

Adresse indiquée par HL en sortie :	
	Octet réservé
	Identifiant du fabricant (Voir liste ci-dessous)
	Octet de poids fort de l'adresse des sauts
	Octet de poids faible de l'adresse des sauts
Adresse du début de la table :	Numéro du Slot du périphérique
	Octet réservé
	Identifiant du fabricant (Voir liste ci-dessous)
	Octet de poids fort de l'adresse des sauts
	Octet de poids faible de l'adresse des sauts
Adresse indiquée par HL en entrée :	Numéro du Slot du périphérique

Liste des identifiants de fabricant :

0 = ASCII	9 = Mitsubishi	18 = Spectravideo
1 = Microsoft	10 = Nippon Denki	19 = Toshiba
2 = Canon	11 = Yamaha	20 = Mitsumi
3 = Casio	12 = Victor	21 = Telematika
4 = Fujitsu	13 = Philips	22 = Gradiente
5 = General	14 = Pioneer	23 = Sharp Epcom
6 = Hitachi	15 = Sanyo	24 = Goldstar
7 = Kyocera	16 = Sharp	25 = Daewoo
8 = National/Panasonic	17 = Sony	26 = Samsung

Fonction 1

Rôle : Nombre de lecteurs installés.

Entrée : Rien.

Sortie : B = Numéro du Slot suivant.

HL = Pointeur sur la zone de travail suivante.

Modifie : ?.

Fonction 2

Rôle : Désactivation des interruptions.

Entrée : Rien.

Sortie : Rien.

Modifie : Rien.

Fonction 3

Rôle : Activation des interruptions.

Entrée : Rien.

Sortie : Rien.

Modifie : Rien.

4 Les variables système et zones de travail

4.1 Introduction aux variables système

L'utilité des variables système n'est plus à démontrer. La plupart des trucs et astuces sont basés sur une bonne connaissance de celles-ci. Il devient ainsi facile de combler les quelques lacunes du Basic. Quant au programmeur en langage machine, il peut parfois ignorer les variables système et utiliser cette zone pour constituer une zone de travail appréciable pour un très gros programme mais il ne faut pas négliger que les Bios utilisent cette zone et ce, particulièrement si le programmeur souhaite développer des routines en langage machine qui coexistent avec l'interpréteur Basic. Dans ce cas, il devra être particulièrement attentif à cette zone mémoire.

4.2 La liste des variables et des zones de travail système

Adresse	Nom	Long.	Fonction
0F380h	RDPRIM	5	Sous-routine de lecture dans un Slot primaire. (Utilisée par la routine RDSLT en Main-ROM).
0F385h	WRPRIM	7	Sous-routine d'écriture dans un Slot primaire. (Utilisée par la routine WRSLT en Main-ROM).
0F38Ch	CLPRIM	14	Sous-routine d'appel dans un Slot primaire. (Utilisée par la routine CALSLT en Main-ROM).
0F39Ah	USRTAB	2	Adresse de la routine définies par l'instruction DEFUSR0=. (Contient l'adresse de la routine d'erreur du Basic par défaut.)
0F39Ch	USRTAB1	2	Adresse de la routine définies par l'instruction DEFUSR1=. (Contient l'adresse de la routine d'erreur du Basic par défaut.)
0F39Eh	USRTAB2	2	Adresse de la routine définies par l'instruction DEFUSR2=. (Contient l'adresse de la routine d'erreur du Basic par défaut.)
0F3A0h	USRTAB3	2	Adresse de la routine définies par l'instruction DEFUSR3=. (Contient l'adresse de la routine d'erreur du Basic par défaut.)
0F3A2h	USRTAB4	2	Adresse de la routine définies par l'instruction DEFUSR4=. (Contient l'adresse de la routine d'erreur du Basic par défaut.)
0F3A4h	USRTAB5	2	Adresse de la routine définies par l'instruction DEFUSR5=. (Contient l'adresse de la routine d'erreur du Basic par défaut.)
0F3A6h	USRTAB6	2	Adresse de la routine définies par l'instruction DEFUSR6=. (Contient l'adresse de la routine d'erreur du Basic par défaut.)
0F3A8h	USRTAB7	2	Adresse de la routine définies par l'instruction DEFUSR7=. (Contient l'adresse de la routine d'erreur du Basic par défaut.)
0F3AAh	USRTAB8	2	Adresse de la routine définies par l'instruction DEFUSR8=. (Contient l'adresse de la routine d'erreur du Basic par défaut.)
0F3ACh	USRTAB9	2	Adresse de la routine définies par l'instruction DEFUSR9=. (Contient l'adresse de la routine d'erreur du Basic par défaut.)

0F3AEh	LINL40	1	Longueur d'une ligne définie par l'instruction WIDTH en mode SCREEN 0. (37 par défaut.)
0F3AFh	LINL32	1	Longueur d'une ligne définie par l'instruction WIDTH en mode SCREEN 1. (29 par défaut.)
0F3B0h	LINLEN	1	Longueur de la ligne actuelle.
0F3B1h	CRTCNT	1	Nombre de ligne de la page actuelle. (24 par défaut.)
0F3B2h	CLMLST	1	Valeur utilisée par PRINT. (LINLEN-(LINLEN MOD 14)-14)
0F3B3h	TXTNAM	2	Adresse de la table des caractères en mode SCREEN 0.
0F3B5h	TXTCOL	2	Adresse de la table des couleurs en SCREEN 0. (MSX2~)
0F3B7h	TXTCGP	2	Adresse de la table des formes en mode SCREEN 0.
0F3B9h	TXATTR	2	Inutilisée.
0F3BBh	TXTPAT	2	Inutilisée.
0F3BDh	T32NAM	2	Adresse de la table des caractères en mode SCREEN 1.
0F3BFh	T32COL	2	Adresse de la table des couleurs en mode SCREEN 1.
0F3C1h	T32CGP	2	Adresse de la table des formes en mode SCREEN 1.
0F3C3h	T32ATR	2	Adresse de la table des attributs de Sprites en mode SCREEN 1.
0F3C5h	T32PAT	2	Adresse de la table des formes de Sprites en mode SCREEN 1.
0F3C7h	GRPNAM	2	Adresse de la table des motifs en mode SCREEN 2.
0F3C9h	GRPCOL	2	Adresse de la table des couleurs en mode SCREEN 2.
0F3CBh	GRPCGP	2	Adresse de la table des formes en mode SCREEN 2.
0F3CDh	GRPATR	2	Adresse de la table des attributs de Sprites en mode SCREEN 2.
0F3CFh	GRPPAT	2	Adresse de la table des formes de Sprites en mode SCREEN 2.
0F3D1h	MLTNAM	2	Adresse de la table des caractères en mode SCREEN 3.
0F3D3h	MLTCOL	2	Adresse de la table des couleurs en mode SCREEN 3.
0F3D5h	MLTCGP	2	Adresse de la table des formes en mode SCREEN 3.
0F3D7h	MLATTR	2	Adresse de la table des attributs de Sprites en mode SCREEN 3.
0F3D9h	MLTPAT	2	Adresse de la table des formes de Sprites en mode SCREEN 3.
0F3DBh	CLIKSW	1	Cliquetis des touches. 0 pour désactiver ; 1 pour activer.
0F3DCh	CSRY	1	Ordonnée du curseur.
0F3DDh	CSRX	1	Abscisse du curseur.
0F3DEh	CNSDFG	1	Fonction FNKSB (00C9h en ROM). 0 = Inactive ; 1 = Active.

0F3DFh	RG0SAV	1	Contenu du registre 0 du VDP.
0F3E0h	RG1SAV	1	Contenu du registre 1 du VDP.
0F3E1h	RG2SAV	1	Contenu du registre 2 du VDP.
0F3E2h	RG3SAV	1	Contenu du registre 3 du VDP.
0F3E3h	RG4SAV	1	Contenu du registre 4 du VDP.
0F3E4h	RG5SAV	1	Contenu du registre 5 du VDP.
0F3E5h	RG6SAV	1	Contenu du registre 6 du VDP.
0F3E6h	RG7SAV	1	Contenu du registre 7 du VDP.
0F3E7h	STATFL	1	Contenu du registre de statut 0 du VDP.
0F3E8h	TRGFLG	1	État des boutons des manettes (0 = Bouton pressé) bit 0 = Barre d'espacement ; bit 1~3 = <i>Inutilisés</i> ; bit 4 = Bouton 1 de la manette 1 ; bit 5 = Bouton 2 de la manette 1 ; bit 6 = Bouton 1 de la manette 2 ; bit 7 = Bouton 2 de la manette 2.
0F3E9h	FORCLR	1	Couleur du texte ou du tracé par défaut.
0F3EAh	BAKCLR	1	Couleur de fond par défaut. (Couleur donné à l'avant-plan lors d'un effacement d'écran sauf en SCREEN 0. Dans ce mode, BAKCLR correspond à la couleur de l'arrière-plan)
0F3EBh	BDRCLR	1	Couleur de la bordure par défaut. (Correspond à la couleur de l'arrière-plan sauf en SCREEN 0.)
0F3ECh	MAXUPD	3	Zone de travail de l'instruction CIRCLE. Contient JP 0000h, saut en 0000h, par défaut.
0F3EFh	MINUPD	3	Zone de travail de l'instruction CIRCLE. Contient JP 0000h, saut en 0000h, par défaut.
0F3F2h	ATRBYT	1	Octet d'attribut graphique. (Couleur de tracé des routines graphiques.)
0F3F3h	QUEUES	2	Adresse de la table des queues QUETAB (0F959h par défaut) de l'instruction PLAY.
0F3F5h	FRCNEW	1	Taille de la mémoire de travail de l'instruction PLAY. (255 par défaut)
0F3F6h	SCNCNT	1	Intervalle entre deux scrutations du clavier. Réglé en permanence par le Basic.
0F3F7h	REPCNT	1	Intervalle avant de commencer la fonction de répétition automatique. Réglé en permanence par le Basic. (50 par défaut)
0F3F8h	PUTPNT	2	Adresse du premier emplacement libre dans le Buffer du clavier. (0FBF0h par défaut)
0F3FAh	GETPNT	2	Adresse de la prochaine donnée issue du clavier (dans le Buffer du

clavier).

0F3FCh	CS120	10	Zone de travail pour le lecteur de cassette. (Jusqu'au MSX2+.)
0F406h	LOW	2	Paramètre cassette. Largeur du 0 sur la bande. (Jusqu'au MSX2+.)
0F408h	HIGH	2	Paramètre cassette. Largeur du 1 sur la bande. (jusqu'au MSX2+.)
0F40Ah	HEADER	1	Paramètre cassette. Longueur du signal. (Jusqu'au MSX2+.)
0F40Bh	ASPCT1	2	Rapport d'aspect du cercle tracé par l'instruction CIRCLE.
0F40Dh	ASPCT2	2	Rapport d'aspect du cercle tracé par l'instruction CIRCLE.
0F40Fh	ENDPRG	5	Leurre de fin de programme pour l'instruction RESUME NEXT. (« : » suivit de 4 zéro par défaut.)
0F414h	ERRFLG	1	Numéro de la dernière erreur.
0F415h	LPTPOS	1	Position de la tête de l'imprimante. 0 au départ.
0F416h	PRTFLG	1	Indicateur de sortie sur imprimante. 0 pas de sortie.
0F417h	NTMSXP	1	Indicateur de type d'imprimante. 0 = Imprimante pour MSX ; 1 = Autre.
0F418h	RAWPRT	1	Mode d'impression. 0 = Impression de caractères ; 1 = Imprimer en mode « Raw ».
0F419h	VLZADR	2	Adresse du caractère remplacé par l'instruction VAL du Basic.
0F41Bh	VLZDAT	1	Caractère remplacé par 0 avec l'instruction VAL du Basic.
0F41Ch	CURLIN	2	Numéro de la ligne en cours d'exécution.
0F41Fh	KBUF	318	« Crunch Buffer » utilisé par le Basic.
0F55Dh	BUFMIN	1	Petit Buffer intermédiaire.
0F55Eh	BUF	258	Buffer du mode direct du Basic.
0F660h	ENDBUF	1	Indicateur de dépassement du Buffer BUF.
0F661h	TTYPOS	1	Position finale du curseur stockée par l'instruction PRINT.
0F662h	DIMFLG	1	Indicateur de l'instruction DIM pour savoir si un tableau est en cours.
0F663h	VALTYP	1	Type de variable contenue dans DAC (0F7F6h). 2 = Entier ; 3 = Chaine de caractères ; 4 = Simple précision ; 8 = Double précision.
0F664h	OPRTYP	1	Le numéro de l'opérateur est stocké momentanément ici lorsqu'une instruction utilise un opérateur.
"	DORES	1	Indicateur de l'interpréteur Basic pour indiquer si il peut compresser des mots-clés ou pas. (Par exemple : les données de DATA ou PRINT ne doivent pas être comprimées.)
0F665h	DONUM	1	Indicateur utilisé pour la compression des nombres.
0F666h	CONTXT	2	Sauvegarde temporaire du pointeur de texte.
0F668h	CONSAV	1	Sauvegarde temporaire du code de l'instruction en cours.
0F669h	CONTYP	1	Type de constante trouvé par l'interpréteur BASIC. Utilisé aussi par la routine CHRGT (00010h en Main-ROM).

0F66Ah	CONLO	8	Valeur de la constante trouvée par l'interpréteur BASIC. Utilisé aussi par la routine CHRGTR (00010h en Main-ROM).
0F672h	MEMSIZ	2	Adresse de la dernière case mémoire disponible sous Basic. Modifié par l'instruction CLEAR.
0F674h	STKTOP	2	Adresse de la fin de la zone réservé pour la pile. Modifié par CLEAR du Basic.
0F676h	TXTTAB	2	Adresse de début des programmes Basic.
0F678h	TEMPPT	2	Pointeur du Basic.
0F67Ah	TEMPST	30	Stockage provisoire.
0F698h	DSCTMP	3	Adresse du premier octet libre dans la table des chaînes de caractères.
0F69Bh	FRETOP	2	Adresse de fin de table de chaînes de caractères.
0F69Dh	TEMP3	2	Mémoire de travail temporaire.
0F69Fh	TEMP8	2	Mémoire de travail temporaire.
0F6A1h	ENDFOR	2	Sauvegarde de la position derrière l'instruction FOR d'une boucle afin de pouvoir revenir avec NEXT.
0F6A3h	DATLIN	2	Numéro de ligne du DATA en cours. Utilisé par READ du Basic.
0F6A5h	SUBFLG	1	Indicateur pour un tableau de l'instruction DEFUSR du Basic.
0F6A6h	FLGINP	1	Indicateur pour les instructions INPUT et READ du Basic.
0F6A7h	TEMP	2	Mémoire de travail pour les instructions INPUT, FOR...NEXT et LET mais aussi ^C.
0F6A9h	PTRFLG	1	Indicateur de conversion de numéro de ligne en pointeur pour l'interpréteur Basic. 0 = Ne pas convertir ; 1 = Convertir.
0F6AAh	AUTFLG	1	Indicateur pour l'instruction AUTO du Basic. 0 = Manuelle ; 1 = Numérotation des lignes automatique en cours.
0F6ABh	AUTLIN	2	Numérotation de ligne actuelle en mode AUTO.
0F6ADh	AUTINC	2	Valeur à incrémenter en mode AUTO.
0F6AFh	SAVTXT	2	Sauvegarde du pointeur de texte après une erreur pour l'instruction RESUME du Basic.
0F6B1h	SAVSTK	2	Sauvegarde de l'adresse de la pile après une erreur afin d'être dans la bonne situation au moment du RESUME (Basic).
0F6B3h	ERRLIN	2	Numéro de la ligne à laquelle la dernière erreur s'est produite.
0F6B5h	DOT	2	Numéro de ligne actuelle. Utilisé par l'instruction LIST du Basic.
0F6B7h	ERRTXT	2	Pointeur de texte utilisé par l'instruction RESUME du Basic.
0F6B9h	ONELIN	2	Numéro de ligne définie par l'instruction ON ERROR GOTO.
0F6BAh	ONEFLG	1	Indicateur d'erreur. 0 mode normal ; 1 traitement d'erreur en cours.
0F6BCh	TEMP2	2	Mémoire de travail pour le programme d'évaluation de formules.
0F6BEh	OLDLIN	2	Numéro de ligne gardé après l'exécution de l'instruction STOP ou END, ou bien un ^C.
0F6C0h	OLDTXT	2	Ancien pointeur de texte. Le pointeur est dirigé sur l'instruction suivant celle où s'est produit l'arrêt.

0F6C2h	VARTAB	2	Pointeur sur le début des variables simples.
0F6C4h	ARYTAB	2	Pointeur sur le début du premier tableaux de variables.
0F6C6h	STREND	2	Adresse de la fin de la zone des variables.
0F6C8h	DATPTR	2	Pointeur de la donnée à lire de l'instruction DATA. Modifié par l'instruction RESTORE.
0F6CAh	DEFTBL	26	Type de variable défini par les instructions DEFSTR, DEFINT, DEFSNG et DEFDBL du Basic pour chacune des 26 lettres de l'alphabet. 2 = Entier ; 4 = Chaîne ; 6 = Simple précision ; 8 = Double précision.
0F6E4h	PRMSTK	2	Pointeur vers le bloc défini précédemment. (Afin de nettoyer les déchets.)
0F6E6h	PRMLEN	2	Nombre d'octets actuellement utilisés.
0F6E8h	PARM1	100	Définition des paramètres.
0F74Ch	PRMPRV	2	Pointeur pour le bloc de paramètres précédent.
0F74Eh	PRMLN2	2	Taille du bloc de paramètres en cours d'élaboration.
0F750h	PARM2	100	Zone pour sauvegarder les blocs en cours de création.
0F7B4h	PRMFLG	1	Indicateur pour savoir si PARM1 a été fouillé.
0F7B5h	ARYTA2	2	Point d'arrêt pour une recherche simple.
0F7B7h	NOFUNS	1	Indicateur de fonction. 0 si aucune fonction n'est active.
0F7B8h	TEMP9	2	Pointeur de stockage temporaire. (Afin de nettoyer les déchets.)
0F7BAh	FUNACT	2	Nombre de fonctions actives.
0F7BCh	SWPTMP	8	Mémoire de travail pour l'instruction SWAP du Basic.
0F7C4h	TRCFLG	1	Indicateur de tracé. 0 = Pas de traçage en cours ; Autre valeur = Traçage en cours.
0F7C5h	FBUFFR	43	Buffer utilisé par les routines mathématiques.
0F7F0h	DECTMP	2	Mémoire de travail.
0F7F2h	DECTM2	2	Mémoire de travail pour les divisions.
0F7F4h	DECCNT	1	Mémoire de travail pour les divisions.
0F7F6h	DAC	16	Accumulateur décimal.
0F806h	HOLD8	48	Sauvegarde temporaire pendant les multiplications décimales.
0F836h	HOLD2	8	Idem ci-dessus.
0F83Eh	HOLD	8	Idem ci-dessus.
0F847h	ARG	16	Accumulateur décimal secondaire.
0F857h	RNDX	8	Dernier nombre aléatoire généré.
0F87Fh	FNKSTR	160	Contient les caractères des touches de fonction à afficher. (16 octets par touche.)
0F91Fh	CGPNT	3	Emplacement de la forme du caractère en ROM. (Slot + adresse)
0F922h	NAMBAS	2	Adresse de la table des caractères ou Bitmap actuelle.

0F924h	CGPBAS	2	Adresse de la table des formes actuelle.
0F926h	PATBAS	2	Adresse de la table des formes de Sprites actuelle.
0F928h	ATRBAS	2	Adresse de la table des attributs de Sprites actuelle.
0F92Ah	CLOC	2	Adresse en VRAM du curseur graphique (SCREEN 2 à 4), Abscisse du curseur graphique (SCREEN 5 à 8)
0F92Ch	CMASK	1	Masque du curseur graphique (SCREEN 2 à 4), Ordonnée du curseur graphique (SCREEN 5 à 8)
0F92Dh	MINDEL	2	Mémoire de travail pour l'instruction LINE du Basic.
0F92Fh	MAXDEL	2	Mémoire de travail pour l'instruction LINE du Basic.
0F931h	ASPECT	2	Aspect du cercle à tracer. Défini par l'angle de l'instruction CIRCLE du Basic.
0F933h	CENCNT	2	Mémoire de travail pour l'instruction CIRCLE du Basic.
0F935h	CLINEF	1	Indicateur pour tracer ou non une ligne vers le centre. Défini par l'angle de l'instruction CIRCLE du Basic.
0F936h	CNPNTS	2	Nombre de points à placer dans un segment de 45°. Utilisé par l'instruction CIRCLE du Basic.
0F938h	CPLOTF	1	Indicateur de polarité. Utilisé par l'instruction CIRCLE du Basic.
0F939h	CPCNT	2	Nombre de points dans 1/8 du cercle. Utilisé par l'instruction CIRCLE du Basic.
0F93Bh	CPCNT8	2	Nombre de points dans le cercle. Utilisé par l'instruction CIRCLE.
0F93Dh	CRCSUM	2	Somme du cercle. Utilisé par l'instruction CIRCLE du Basic.
0F93Fh	CSTCNT	2	Variable pour maintenir le nombre de points de l'angle de départ. Utilisé par l'instruction CIRCLE du Basic.
0F941h	CSCLXY	1	Rapport entre la largeur et la hauteur. (CIRCLE)
0F942h	CSAVEA	2	Adresse du premier pixel de couleur différente. Utilisé par l'instruction PAINT du Basic.
0F944h	CSAVEM	1	Masque du premier pixel de couleur différente. Utilisé par l'instruction PAINT du Basic.
0F945h	CXOFF	2	Décalage horizontal depuis la position sauvegardée. Utilisé par l'instruction PAINT du Basic.
0F947h	CYOFF	2	Décalage vertical depuis la position sauvegardée. Utilisé par l'instruction PAINT du Basic.
0F949h	LOHMSK	1	Garde la position la plus à gauche d'une excursion de LH. Utilisé par l'instruction PAINT du Basic.
0F94Ah	LOHDIR	1	Garde la nouvelle direction de peinture requise par une excursion de LH. Utilisé par l'instruction PAINT du Basic.
0F94Bh	LOHADR	2	Garde la position la plus à gauche d'un LH. Utilisé par l'instruction PAINT du Basic.
0F94Dh	LOHCNT	2	Garde la taille d'une excursion de LH. Utilisé par l'instruction PAINT du Basic.
0F94Fh	SKPCNT	2	Garde le nombre de saut retourné par la routine SCANR du Bios. Utilisé par l'instruction PAINT du Basic.

0F951h	MOVCNT	2	Garde le nombre de déplacement retourné par la routine SCANR du Bios. Utilisé par l'instruction PAINT du Basic.
0F953h	PDIREC	1	Direction dans laquelle on peint. Utilisé par l'instruction PAINT. 40h = Vers le bas, C0h = Vers le haut, 00h = Terminé.
0F954h	LFPROG	1	Mis à 1 lors d'une progression vers la gauche. Utilisé par l'instruction PAINT du Basic.
0F955h	RTPROG	1	Mis à 1 lors d'une progression vers la droite. Utilisé par l'instruction PAINT du Basic.
0F956h	MCLTAB	2	Pointeur vers la chaîne de caractères du Macro langage de l'instruction PLAY ou DRAW du Basic.
0F958h	MCLFLG	1	Indicateur indiquant si la chaîne de caractères du langage macro appartient à PLAY (>0) ou à DRAW (0).
0F959h	QUETAB	24	<p>Table des queues de PLAY et de l'interface RS-232.</p> <pre> QUETAB: DEFB 00H ; AQ Position de départ DEFB 00H ; AQ Indicateur de position DEFB 00H ; AQ Indicateur de replacement DEFB 7FH ; AQ Taille DEFW VOICAQ ; AQ Adresse ; DEFB 00H ; BQ Position de départ DEFB 00H ; BQ Indicateur de position DEFB 00H ; BQ Indicateur de replacement DEFB 7FH ; BQ Taille DEFW VOICBQ ; BQ Adresse ; DEFB 00H ; CQ Position de départ DEFB 00H ; CQ Indicateur de position DEFB 00H ; CQ Indicateur de replacement DEFB 7FH ; CQ Taille DEFW VOICCQ ; CQ Adresse ; DEFB 00H ; RQ Position de départ DEFB 00H ; RQ Indicateur de position DEFB 00H ; RQ Indicateur de replacement DEFB 00H ; RQ Taille DEFW 0000H ; RQ Adresse </pre> <p>Les trois tables de contrôle de la musique sont initialisées par la routine GICINI (00090h) et ensuite géré par la routine d'interruption et la routine PUTQ (000F9h). Cette table de commande RS232 n'est plus utilisé depuis le MSX2. (Obsolète).</p>
0F971h	QUEBAK	4	<p>Table de caractères de remplacement des queues.</p> <pre> QUEBAK: DEFB 00H ; AQ Caractère de remplacement DEFB 00H ; BQ Caractère de remplacement DEFB 00H ; CQ Caractère de remplacement DEFB 00H ; RQ Caractère de remplacement </pre>
0F975h	VOICAQ	128	Queue musicale pour la voix A.
0F9F5h	VOICBQ	128	Queue musicale pour la voix B
0FA75h	VOICCQ	128	Queue musicale pour la voix C

0FAF5h	RS2IQ	64	Queue de la RS-232. (Retiré du standard MSX)
0FAF5h	DPPAGE	1	Page affichée. (MSX2~)
0FAF6h	ACPAGE	1	Page active. (MSX2~)
0FAF7h	AVCSAV	1	Sauvegarde du port de contrôle AV. (MSX2~)
0FAF8h	EXBRSA	1	Slot dans lequel se trouve la Sub-ROM. (MSX2~)
0FAF9h	CHRCNT	1	Compteur de caractère dans le Buffer de conversion Roma-Kana.
0FAFAh	ROMA	2	Sauvegarde le caractère lors d'un conversion Roma-Kana.
0FAFCh	MODE	1	Indicateur du mode conversion Roma-Kana et de taille de la VRAM. bit 0 = Conversion. bit 1~2 = Taille de la VRAM. 00 pour 16Ko ; 01 pour 64Ko ; 10 pour 128Ko ; 11 pour 192Ko. bit 3 = 1 pour appliquer le masque. (SCREEN 0 ~ 3). bit 4~6 = 0. bit 7 = 1 Katakana ; 0 Hiragana.
0FAFDh	NORUSE	1	Utilisé par le pilote « Kanji Driver ». bit 0~2 = Opérateur logique utilisé pour l'affichage des Kanji. 000 pour IMP ; 001 pour AND ; 010 pour OR ; 011 pour XOR ; 100 pour NOT. bit 3 = 1 pour couleur 0 transparente. bit 4 = <i>Inutilisé</i> . bit 5 = bit 6 = Permet le défilement avec SHIFT+Haut/Bas (mode Kanji). bit 7 = Retour au mode texte interdit.
0FAFEh	XSAVE	2	Sauvegarde de l'abscisse du curseur graphique lors de l'utilisation du crayon optique. Le bit 15 indique une demande d'interruption. Les bits 14~8 sont à zéro. (MSX2~)
0FB00h	YSAVE	2	Sauvegarde de l'ordonnée du curseur graphique lors de l'utilisation du crayon optique. Le bit 15 indique une demande d'interruption. Les bits 14~8 sont à zéro. (MSX2~)
0FB02h	LOGOPR	1	Opérateur logique à effectuer lors d'un tracé ou d'une copie graphique : 0 = IMP ; 1 = AND ; 2 = OR ; 3 = XOR ; 4 = NOT.
0FB03h	TOCNT	1	Utilisée par l'interface RS-232C. (Retiré du standard MSX)
0FB04h	RSFCB	2	Adresse du FCB (« File Control Block ») de la RS-232C. (Retiré du standard MSX)
0FB06h	RSIQLN	1	Utilisée par l'interface RS-232C. (Retiré du standard MSX)
0FB07h	MEXBIH	5	RS-232C. (Retiré du standard MSX)

			FB07h+0 : RST 30h (0F7h) FB07h+1 : byte data FB07h+2 : (Low) FB07h+3 : (High) FB07h+4 : RET (0C9h)
0FB0Ch	OLDSTT	5	RS-232C. (Retiré du standard MSX) FB0Ch+0 : RST 30h (0F7h) FB0Ch+1 : byte data FB0Ch+2 : (Low) FB0Ch+3 : (High) FB0Ch+4 : RET (0C9h)
0FB12h	OLDINT	5	RS-232C. (Retiré du standard MSX) FB12h+0 : RST 30h (0F7h) FB12h+1 : byte data FB12h+2 : (Low) FB12h+3 : (High) FB12h+4 : RET (0C9h)
0FB17h	DEVNUM	1	Utilisée par l'interface RS-232C. (Retiré du standard MSX)
0FB18h	DATCNT	3	RS-232. (Retiré du standard MSX) FB18h +0 : byte data FB18h : adresse
0FB1Bh	ERRORS	1	Utilisée par l'interface RS-232C. (Retiré du standard MSX)
0FB1Ch	FLAGS	1	Utilisée par l'interface RS-232C. (Retiré du standard MSX)
0FB1Dh	ESTBLS	1	Utilisée par l'interface RS-232C. (Retiré du standard MSX)
0FB1Eh	COMMSK	1	Utilisée par l'interface RS-232C. (Retiré du standard MSX)
0FB1Fh	LSTCOM	1	Utilisée par l'interface RS-232C. (Retiré du standard MSX)
0FB20h	LSTMOD	1	Utilisée par l'interface RS-232C. (Retiré du standard MSX)
0FB20h	HOKVLD	1	Le bit 0 de cet octet indique la présence d'un Bios d'extension. 0 = Pas de Bios ; 1 = Il y a au moins un Bios qui peut être appelé à l'adresse 0FFCAh (EXTBIO).
0FB35h	PRSCNT	1	Mémoire de travail pour l'instruction PLAY du Basic afin de compter les commandes effectuées.
0FB36h	SAVSP	2	L'instruction PLAY du Basic y sauvegarde le pointeur de pile (contenu du registre SP) lorsqu'elle est exécuter.
0FB38h	VOICEN	1	Nombre de voix jouées actuellement par l'instruction PLAY.
0FB39h	SAVVOL	2	Sauvegarde du volume lors d'un pause.
0FB3Bh	MCLLEN	1	Mémoire de travail pour l'instruction PLAY du Basic.
0FB3Ch	MCLPTR	2	Adresse de la macro en cours d'analyse.
0FB3Eh	QUEUEN	1	Numéro de la queue actuelle.
0FB3Fh	MUSICF	1	Indicateur d'interruption musicale.
0FB40h	PLYCNT	1	Nombre de chaînes macro dans la file d'attente de PLAY.

0FB41h	VCBA	37	Données pour la voix 0.
0FB66h	VCBB	37	Données pour la voix 1.
0FB8Bh	VCBC	37	Données pour la voix 2.
0FBB0h	ENSTOP	1	Indicateur différent de 0 lorsqu'il est possible d'interrompre l'exécution d'un programme Basic en pressant les touches CTRL, SHIFT, GRPH et KANA/CODE simultanément.
0FBB1h	BASROM	1	Programme Basic en ROM si différent de 0. (L'exécution du programme ne peut être interrompue par CTRL+STOP dans ce cas.)
0FBB2h	LINTTB	24	Table de 24 indicateurs de fin de ligne pour l'interpréteur Basic. 0 = La ligne correspondante contient une ligne de programme qui continue sur la ligne suivante ; Autre valeur = La ligne correspondante contient une ligne de programme qui se termine ici.
0FBCAh	FSTPOS	2	Pointeur vers le premier caractère entré par les routines INLIN (00B1h) et QINLIN (00B4h) en Main-ROM.
0FBCCh	CODSAV	1	Contient le code du caractère placé sous le curseur.
0FBCDh	FNKSWI	1	Indique quelle série de touches de fonction est affichée. 0 = Touches F1 à F6 ; Autre valeur = F6 à F10.
0FBCEh	FNKFLG	10	Dix indicateurs pour l'instruction KEY(x) ON/OFF/STOP du Basic. 0 = KEY(x) OFF/STOP ; Autre valeur = KEY(x) ON.
0FBD8h	ONGSBF	1	Indicateur pour l'instruction ON... GOSUB du Basic.
0FBD9h	CLIKFL	1	Indicateur pour savoir si le click a déjà eu lieu.
0FBDAh	OLDKEY	11	Statut précédent de chaque ligne de la matrice du clavier.
0FBE5h	NEWKEY	11	Nouveau statut de chaque ligne de la matrice du clavier. Le statut est actualisé par la routine d'interruption KEYINT.
0FBF0h	KEYBUF	40	Buffer du clavier par défaut. Contient la valeur des dernières touches pressées.
0FC18h	LINWRK	40	Zone de travail pour la gestion de l'écran.
0FC40h	PATWRK	8	Zone de travail pour le convertisseur noms vers formes.
0FC48h	BOTTOM	2	Adresse du début de la zone de RAM disponible.
0FC4Ah	HIMEM	2	Adresse de la fin de la zone de RAM disponible.
0FC4Ch	TRPTBL	78	Table composée de 3 octets pour chacune des instructions suivantes : FC4Ch~FC69h (3 × 10 octets) = ON KEY GOSUB FC6Ah~FC6Ch (3 × 1 octet) = ON STOP GOSUB FC6Dh~FC6Fh (3 × 1 octet) = ON SPRITE GOSUB FC70h~FC7Eh (3 × 5 octets) = ON GOSUB STRIG FC7Fh~FC81h (3 × 1 octet) = ON INTERVAL GOSUB FC82h~FC99h - Réservé Le premier octet sert d'indicateur. 0 = OFF ; 1 = ON ; 2 = STOP ; 3 = Appel en cours. 7 = Appel en attente. Les 2 autres octets contiennent l'adresse du numéro de ligne de la routine à appeler par le GOSUB dans le programme Basic.
0FC9Ah	RTYCNT	1	Contrôle d'interruption.

0FC9Bh	INTFLG	1	Indicateur d'interruption.
0FC9Ch	PADX	1	Valeur en X de la molette (« paddle controller »).
0FC9Dh	PADY	1	Valeur en Y de la molette (« paddle controller »).
0FC9Eh	JIFFY	2	Compteur croissant de l'instruction TIME du Basic.
0FCA0h	INTVAL	2	Valeur de l'intervalle de l'instruction ON INTERVAL=... GOSUB du Basic.
0FCA2h	INTCNT	2	Compteur décroissant de l'instruction ON INTERVAL=... GOSUB.
0FCA4h	LOWLIM	1	Utilisé durant la lecture cassette.
0FCA5h	WINWID	1	Idem ci-dessus.
0FCA6h	GRPHED	1	En-tête pour la sortie des caractères graphiques.
0FCA7h	ESCCNT	1	Compteur pour une séquence d'escape.
0FCA8h	INSFLG	1	Indicateur de mode insertion.
0FCA9h	CSRSW	1	« Cursor display switch » : Indicateur pour le curseur.
0FCAAh	CSTYLE	1	Forme du curseur. 0 = le curseur entier ; autre valeur = un petit curseur. (mode insertion)
0FCABh	CAPST	1	Indicateur CAPS d'entrée de caractère. 0 = Caractère en minuscules ; autre valeur = majuscules.
0FCACH	KANAST	1	Indicateur de mode kana. (MSX Japonais)
0FCADh	KANAMD	1	Indicateur spécifique au mode kana. (MSX Japonais)
0FCAEh	FLBMEM	1	Indicateur de chargement : 0 si un programme Basic se charge.
0FCAFh	SCRMOD	1	Mode d'écran actuel.
0FCB0h	OLDSCR	1	Ancien mode d'écran. Utilisé pour savoir dans quel mode texte il faut repasser après un mode graphique.
0FCB1h	CASPRV	1	Mémoire de travail pour la cassette. Sur les MSX turbo R, le bit 7 de cet octet contient la valeur du bit 7 écrite au port 0A7h. (Bit 7 à 1 pour LED de la touche Pause allumée)
0FCB2h	BRDATR	1	Couleur de la frontière pour l'instruction PAINT du Basic.
0FCB3h	GXPOS	2	Abscisse du curseur graphique.
0FCB5h	GYPOS	2	Ordonnée du curseur graphique.
0FCB7h	GRPACX	2	Accumulateur graphique X.
0FCB9h	GRPACY	2	Accumulateur graphique Y.
0FCBBh	DRWFLG	1	Indicateur pour l'instruction DRAW du Basic.
0FCBCh	DRWSCL	1	Taille du zoom pour l'instruction DRAW du Basic.
0FCBDh	DRWANG	1	Angle pour l'instruction « DRAW » du Basic
0FCBEh	RUNBNF	1	Indicateur d'Entrée/Sortie : 255 = E/S binaire en cours
0FCC1h	EXPTBL	4	Un Indicateur pour chacun des Slots primaires : le bit 7 est à 1 lorsque le Slot est étendu en Slot secondaires. Les autres bits restent à 0. Par conséquent, 0FCC1h (appelé aussi MNROM) indiquera le Slot de la Main-ROM (0 ou 0-0).

0FCC5h	SLTTBL	4	Indicateur du Slot secondaire actuel pour chaque Slot primaire. bit 0 et 1 = Slot secondaire sélectionné dans le Slot primaire 0, bit 2 et 3 = Slot secondaire sélectionné dans le Slot primaire 1, bit 4 et 5 = Slot secondaire sélectionné dans le Slot primaire 2, bit 6 et 7 = Slot secondaire sélectionné dans le Slot primaire 3.
0FCC9h	SLTATR	64	Attributs de chaque Slot (voir le chapitre « Utiliser les Slots »)
0FD09h	SLTWRK	128	Zone de travail pour chaque Slot
0FD89h	PROCNM	16	Nom des instructions supplémentaires exécutables par CALL.
0FD99h	DEVICE	1	Identifiant « ID device » pour la cartouche (0-3)
0FFE7h	RG8SAV	1	Contenu du registre 8 du VDP. (MSX2~)
0FFE8h	RG9SAV	1	Contenu du registre 9 du VDP. (MSX2~)
0FFE9h	RG10SAV	1	Contenu du registre 10 du VDP. (MSX2~)
0FFEAh	RG11SAV	1	Contenu du registre 11 du VDP. (MSX2~)
0FFEBh	RG12SAV	1	Contenu du registre 12 du VDP. (MSX2~)
0FFEC	RG13SAV	1	Contenu du registre 13 du VDP. (MSX2~)
0FFEDh	RG14SAV	1	Contenu du registre 14 du VDP. (MSX2~)
0FFEEh	RG15SAV	1	Contenu du registre 15 du VDP. (MSX2~)
0FFEFh	RG16SAV	1	Contenu du registre 16 du VDP. (MSX2~)
0FFF0h	RG17SAV	1	Contenu du registre 17 du VDP. (MSX2~)
0FFF1h	RG18SAV	1	Contenu du registre 18 du VDP. (MSX2~)
0FFF2h	RG19SAV	1	Contenu du registre 19 du VDP. (MSX2~)
0FFF3h	RG20SAV	1	Contenu du registre 20 du VDP. (MSX2~)
0FFF4h	RG21SAV	1	Contenu du registre 21 du VDP. (MSX2~)
0FFF5h	RG22SAV	1	Contenu du registre 22 du VDP. (MSX2~)
0FFF6h	RG23SAV	1	Contenu du registre 23 du VDP. (MSX2~)
0FFF7h	ROMSLT	1	Contient le numéro de Slot de la Main-ROM. (MSX2~) Cette variable a été introduite avec l'MSX2 mais n'a pratiquement jamais été utilisée. C'est même indiqué de ne pas l'utiliser dans le MSX Datapack. La Main-ROM est forcément placée dans le Slot 0 ou 0-0. De ce fait, pour déterminer le numéro de Slot de la Main- ROM, veuillez utiliser EXPTBL (0FCC1H).
0FFF8h		2	Réservé.
0FFFAh	RG25SAV	1	Contenu du registre 25 du VDP. (MSX2+~)
0FFFBh	RG26SAV	1	Contenu du registre 26 du VDP. (MSX2+~)
0FFFCh	RG27SAV	1	Contenu du registre 27 du VDP. (MSX2+~)
0FFFDh		2	Sauvegarde temporaire du registre SP.

0FFFFh SLTSL 1 Adresse d'accès au registre de sélection des Slots secondaires (du Slot primaire sélectionné). La valeur lue correspond aux Slots secondaires dont les bits sont inversés afin de pouvoir déterminer si le Slot est étendu ou pas.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
$\overline{SS3}$	$\overline{SS2}$	$\overline{SS1}$	$\overline{SS0}$				

Attention : Étant donné qu'il faut sélectionner le Slot primaire correspondant avant de pouvoir sélectionner un Slot secondaire, prenez soin de désactiver les interruptions jusqu'à ce que les variables du système soit remises sur sa plage mémoire.

4.3 Quelques exemples d'utilisation des variables système

- Sous Basic, vous désirez couper un bout d'écran afin de réserver quelques lignes (sur l'exemple des touches de fonction). Il vous suffit de faire :

```
POKE &HF3B1, 24-nn          (nn étant le nombre de lignes à réserver)
```

- Il est parfois utile de connaître la largeur de l'écran (réglé avec « WIDTH ») dans un programme d'application. Essayez :

```
L=PEEK (&HF3B0)
```

- Dans le même ordre d'idées, le nombre maximum de fichiers ouvrables (défini par l'instruction « MAXFILES ») s'obtient par :

```
M=PEEK (&HF85F)
```

- Qui n'a jamais connu le problème de Buffer du clavier qui renvoie des caractères au mauvais moment. Effacer ce Buffer est souvent utile :

En Basic :

```
POKE &HF3FA, PEEK (&HF3F8)
POKE &HF3FB, PEEK (&HF3F9)
```

En assembleur :

```
LD HL, (PUTPNT)
LD (GETPNT), HL
```

- Voici un truc un peu plus sophistiqué : l'interdiction de l'action combinée des touches CTRL+STOP peut facilement être obtenue en faisant croire au MSX que votre programme se trouve en cartouche de mémoire morte.

```
POKE &HFBB1, 1 fera parfaitement l'affaire.
```

- Au contraire, pour ceux qui désirent accéder à des programmes Basic protégés contre le CTRL + STOP, il leur suffit de taper :

```
POKE &HFBB0, 1 avant de charger le programme.
```

Puis, le programme ayant démarré, il faut presser simultanément les touches CTRL, SHIFT, GRAPH et CODE. Le MSX rendra alors la main.

- Pour ceux que la présence du curseur rassure, ils peuvent essayer en mode direct :

```
POKE &HFCA9,1
```

Pour se rendre compte de l'effet obtenu, il suffit de rentrer le petit programme suivant :

```
10 PRINT"Hello"  
20 GOTO 20
```

et d'entrer RUN.

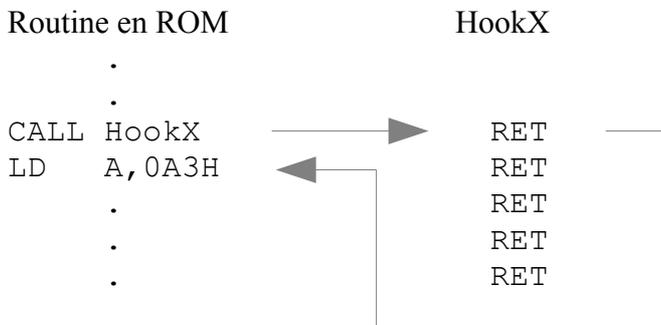
- Le dernier exemple est un peu plus conséquent que les autres. Il s'agit d'utiliser la zone réservée à la définition des touches fonction au mieux de manière à pouvoir assigner n'importe quelle chaîne de caractères à une touche. Notons qu'on peut mettre au maximum 39 caractères par touche, sachant que l'on ne dispose de toutes manières que de 160 octets et que si l'on étend les contenu d'une touche de fonction, ce ne peut être qu'aux dépens de la suivante.

Le petit programme Basic suivant illustre cet exemple :

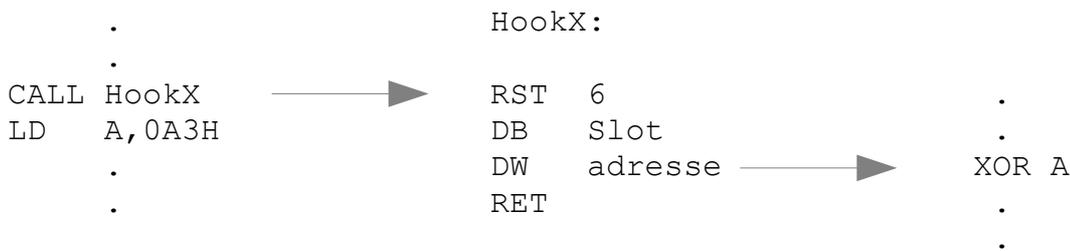
```
10 AD=&HF87F:CLS  
20 READ A$:IF LEN(A$)>39 THEN PRINT"Impossible!":END  
30 FOR I=1 TO LEN(A$)  
40 I$=MID$(A$,I,1)  
50 POKE AD-1+I,ASC(I$)  
60 NEXT I  
70 POKE AD-1+I,0:END  
100 DATA "C'est un peu long jeune homme ..."
```

5 Les Hooks

Un Hook est une zone mémoire de cinq octets en mémoire vive (RAM) qui permet d'étendre ou dérouter les fonctions en mémoire morte. Au départ, les cinq octets contiennent tous 0C9h (RET), comme dans l'exemple suivant :



Cependant, il est facile de modifier les 5 octets afin de détourner la fonction en ROM, comme ceci :



Pour illustrer le fonctionnement des Hooks, je vous propose un exemple en Basic qui utilise le Hook H.LIST pour empêcher l'accès à la liste du programme.

```
10 POKE &HFF8C, &H40      ' met dans H.LIST :
20 POKE &HFF8B, &H9B      ' POP HL
30 POKE &HFF8A, &HC3      ' JP 0409BH
40 POKE &HFF89, &HE1      ' <- remplacer le &HC9 en dernier
```

Il suffit de faire `POKE &HFF89, &HC9` pour restituer son pouvoir à la commande LIST.

5.1 La liste des Hooks

Voici la liste des Hooks pour les MSX1 et MSX2. Elle comprend l'adresse, le nom du Hook, le nom de la routine appelée, ainsi que son utilité :

0FD9AH	H.KEYI
Appel :	Au début de la routine d'interruptions KEYINT (00038h en Main-ROM).
Rôle :	Permet de tester si l'interruption a été provoquée par un dispositif autre que le VDP. (La RS-232C, le MSX-Midi, etc.)

0FD9Fh	<i>H.TIMI</i>
Appel :	Appelé par la routine d'interruptions KEYINT (00038h en Main-ROM).
Rôle :	Permet d'ajouter un Timer, de gérer les interruptions ligne du VDP, la collision de sprites, etc. L'appel à ce Hook se fait juste après la lecture du registre de statut 0 du VDP. Le registre A contiendra la valeur lue lorsque ce Hook est appelé afin de savoir si il s'agit de l'interruption en fin d'affichage (VBI) ou non.
0FDA4h	<i>H.CHPU</i>
Appel :	Au début de la routine CHPUT (000A2h en Main-ROM), sortie d'un caractère à l'écran.
Rôle :	Permet de sortir le caractère sur un périphérique autre que l'écran.
0FDA9h	<i>H.DSPC</i>
Appel :	Au début de la routine interne DSPCSR (« DiSPlay CurSoR ») d'affichage du curseur.
Rôle :	Permet l'accès à d'autres périphériques que l'écran.
0FDAEh	<i>H.ERAC</i>
Appel :	Au début de la routine interne ERACSR (« ERAse CurSoR ») d'effacement du curseur.
Rôle :	Permet l'accès à d'autres périphériques que l'écran.
0FDB3h	<i>H.DSPF</i>
Appel :	Au début de la routine DSPFNK (0011Dh en Main-ROM / Sub-ROM), affichage du contenu des touches de fonction.
Rôle :	Permet l'accès à d'autres périphériques que l'écran.
0FDB8h	<i>H.ERAF</i>
Appel :	Au début de la routine ERAFNK (000CCh en Main-ROM), effacement du contenu des touches de fonction.
Rôle :	Permet l'accès à d'autres périphériques que l'écran.
0FDBDh	<i>H.TOTE</i>
Appel :	Au début de la routine TOTEXT (000D2h en Main-ROM), passage en mode texte.
Rôle :	Permet l'accès à d'autres périphériques que l'écran
0FDC2h	<i>H.CHGE</i>
Appel :	Au début de la routine CHGET (0009Fh en Main-ROM), lecture d'un caractère au clavier.
Rôle :	Permet l'accès à d'autres périphériques d'entrée que le clavier

0FDC7h	<i>H.INIP</i>
Appel :	Au début de la routine interne INIPAT (« INItialize PATtern ») de remplissage de la table des formes en mode texte.
Rôle :	Permet de modifier le jeu de caractères lorsque l'on revient en mode texte. (Après le SCREEN 2 par exemple.)
0FDCCCh	<i>H.KEYC</i>
Appel :	Au début de la routine interne KEYCOD (« KEY CODer ») de lecture de clavier.
Rôle :	Permet d'intercepter la lecture du clavier. Lorsque le Hook H.KEYC est appelé, l'accumulateur contient dix fois le numéro de ligne plus le numéro de colonne de la touche enfoncée dans la matrice clavier
0FDD1h	<i>H.KYEA</i>
Appel :	Au début de la routine interne KYEASY (« KeY EASY ») de conversion d'un caractère lu au clavier.
Rôle :	Permet de modifier la manière dont une touche est interprétée sous Basic.
0FDD6h	<i>H.NMI</i>
Appel :	Au début de la routine NMI (00060h en Main-ROM) d'interruptions non masquables.
Rôle :	Permet le traitement des interruptions non masquables.
0FDDBh	<i>H.PINL</i>
Appel :	Au début de la routine PINLIN (000AEh en Main-ROM) d'entrée d'une ligne de programme au clavier.
Rôle :	Permet d'utiliser les 80 colonnes, ou un autre périphérique d'entrée que le clavier.
0FDE0h	<i>H.QINL</i>
Appel :	Au début de la routine QINLIN (000B4h en Main-ROM) de l'entrée au clavier avec affichage d'un point d'interrogation.
Rôle :	Permet d'utiliser les 80 colonnes, ou un autre périphérique d'entrée que le clavier.
0FDE5h	<i>H.INLI</i>
Appel :	Au début de la routine INLIN (000B1h en Main-ROM) d'entrée au clavier.
Rôle :	Permet d'utiliser les 80 colonnes, ou un autre périphérique d'entrée que le clavier.
0FDEAh	<i>H.ONGO</i>
Appel :	Au début de la routine interne ONGOTP (« ON GOTO Procedure »), pour les instructions Basic de type ON GOTO, ON GOSUB.

Rôle : Permet de détourner ce type d'instructions.

0FDEFh ***H.DSKO***

Appel : Au début de la routine DSKO (Disk-ROM), utilisée par l'instruction DSKO\$ du Disk-Basic.

Rôle : Permet de détourner les écritures de secteur au disque.

0FDF4h ***H.SETS***

Appel : Au début de la routine SETS (00179h en Sub-ROM). Les instructions du Basic précédées par SET appellent cette routine.

Rôle : Détourner l'accès à ces instructions.

0FDF9h ***H.NAME***

Appel : Au début de la routine NAME (Disk-ROM), utilisée par l'instruction MAME du Disk-Basic.

Rôle : Empêcher de renommer les fichiers.

0FDFEh ***H.KILL***

Appel : Au début de la routine KILL (Disk-ROM), utilisée par l'instruction KILL du Disk-Basic.

Rôle : Empêcher d'effacer les fichiers.

0FE03h ***H.IPL***

Appel : Au début de l'instruction IPL (Sub-ROM) de chargement du programme initial.

Rôle : L'instruction IPL du Basic (v.2.0 ou supérieure) ne fait qu'appeler ce Hook afin qu'un programmeur puisse créer sa propre instruction. Voir le chapitre 7.8 « [Ajouter une instruction au Basic avec CMD ou IPL](#) » pour plus de précision.

0FE08h ***H.COPY***

Appel : Au début de la routine COPY (Disk-ROM), utilisée par l'instruction COPY du Disk-Basic.

Rôle : Détourner les copies sur disquettes.

0FE0Dh ***H.CMD***

Appel : Au début de l'instruction CMD (Sub-ROM).

Rôle : L'instruction CMD du Basic (v.2.0 ou supérieure) ne fait qu'appeler ce Hook afin qu'un programmeur puisse créer sa propre instruction. Voir le chapitre 7.8 « [Ajouter une instruction au Basic avec CMD ou IPL](#) » pour plus de précision.

0FE12h ***H.DSKF***

Appel : Au début de la routine DSKF (Disk-ROM), utilisée l'instruction du Disk-Basic.
Rôle : Empêcher de connaître la mémoire disponible sur les disques.

0FE17h ***H.DSKI***

Appel : Au début de la routine DSKI (Disk-ROM), utilisée par l'instruction DSKI\$ du Disk-Basic.

Rôle : Détourné pour obtenir un accès aux lecteurs de disquette.

0FE1Ch ***H.ATTR***

Appel : Au début de la routine ATTR (« ATTRibute »), utilisée par l'instruction ATTR\$ du Disk-Basic..

Rôle : Détourner l'instruction.

0FE21h ***H.LSET***

Appel : Au début de la routine LSET (« Left SET »), utilisée par l'instruction LSET du Disk-Basic.

Rôle : Détourné pour obtenir un accès aux lecteurs de disquette.

0FE26h ***H.RSET***

Appel : Au début de la routine RSET (« Right SET »), utilisée par l'instruction RSET du Disk-Basic.

Rôle : Détourné pour obtenir un accès aux lecteurs de disquette.

0FE2Bh ***H.FIEL***

Appel : Au début de la routine FIELD, utilisée par l'instruction FIELD du Disk-Basic.

Rôle : Détourné pour obtenir un accès aux lecteurs de disquette.

0FE30h ***H.MKI\$***

Appel : Au début de la routine MKI (« MaKe Integer »), utilisée par l'instruction MKI\$ du Disk-Basic.

Rôle : Détourné pour obtenir un accès aux lecteurs de disquette.

0FE35h ***H.MKS\$***

Appel : Au début de la routine MKS (« MaKe Simple precision »), utilisée par l'instruction MKS\$ du Disk-Basic.

Rôle : Détourné pour obtenir un accès aux lecteurs de disquette.

0FE3Ah ***H.MKD\$***

Appel : Au début de la routine MKD (« MaKe Double precision »), utilisée par l'instruction MKD\$ du Disk-Basic.

Rôle : Détourné pour obtenir un accès aux lecteurs de disquette.

0FE3Fh ***H.CVI***

Appel : Au début de la routine CVI (« ConVert Integer »), utilisée par l'instruction CVI du Disk-Basic.

Rôle : Détourné pour obtenir un accès aux lecteurs de disquette

0FE44h ***H.CVS***

Appel : Au début de la routine CVS (« ConVert Single precision »), utilisée par l'instruction CVS du Disk-Basic.

Rôle : Détourné pour obtenir un accès aux lecteurs de disquette.

0FE49h ***H.CVD***

Appel : Au début de la routine CVD (« ConVert Double precision »), utilisé par l'instruction CVD du Disk-Basic.

Rôle : Utilisé par la Disk-ROM.

0FE4Eh ***H.GETP***

Appel : Au début de la routine GETPTR « GET file PoinTeR »), positionnement sur un fichier.

Rôle : Utilisé par la Disk-ROM.

0FE53h ***H.SETF***

Appel : Au début de la routine SETFIL (« SET FILE »), positionnement d'un pointeur sur un fichier ouvert au préalable.

Rôle : Utilisé par la Disk-ROM.

0FE58h ***H.NOFO***

Appel : Au début de la routine NOFOR (« NO FOR clause »), utilisée par l'instruction OPEN du Basic.

Rôle : Utilisé par la Disk-ROM.

0FE5Dh ***H.NULO***

Appel : Au début de la routine NULOPN (« NULI file OPen »), utilisée par les instructions LOAD, KILL, MERGE, etc.

Rôle : Utilisé par la Disk-ROM.

0FE62h	<i>H.NTFL</i>
Appel :	Au début de la routine NTFL0 (« NoT FiLe number 0 »), utilisée par l'instruction CLOSE du Basic.
Rôle :	Utilisé par la Disk-ROM.
0FE67h	<i>H.MERG</i>
Appel :	Au début de la routine MERGE (« MERGE program files »), utilisée par l'instruction MERGE du Basic.
Rôle :	Utilisé par la Disk-ROM.
0FE6Ch	<i>H.SAVE</i>
Appel :	Au début de la routine SAVE, utilisée au début de l'instruction SAVE du Basic.
Rôle :	Utilisé par la Disk-ROM.
0FE71h	<i>H.BINS</i>
Appel :	Au début de la routine BINSAV (« BINary SAve »), sauvegarde d'une zone mémoire en binaire.
Rôle :	Utilisé par la Disk-ROM.
0FE76h	<i>H.BINL</i>
Appel :	Au début de la routine BINLOD (« BINary LOaD »), chargement d'une zone mémoire en binaire.
Rôle :	Utilisé par la Disk-ROM.
0FE7Bh	<i>H.FILE</i>
Appel :	Au début de la routine FILES, utilisée par l'instruction FILES du Disk-Basic.
Rôle :	Utilisé par la Disk-ROM.
0FE80h	<i>H.DGET</i>
Appel :	Au début de la routine DGET (« Disk GET »), utilisée par l'instruction GET du Disk-Basic.
Rôle :	Utilisé par la Disk-ROM.
0FE85h	<i>H.FILO</i>
Appel :	Au début de la routine FILOU1 (« FiLe OUtput 1 »), sortie dans un fichier.
Rôle :	Utilisé par la Disk-ROM.
0FE8Ah	<i>H.INDS</i>
Appel :	Au début de la routine INDSKC (« INput DiSK Character »).

Rôle : Utilisé par la Disk-ROM.

0FE8Fh ***H.RSLF***

Appel : Au début de la routine qui sélectionne l'ancien lecteur de disquettes comme lecteur actuel.

Rôle : Utilisé par la Disk-ROM.

0FE94h ***H.SAVD***

Appel : Au début de la routine de sauvegarde du lecteur actuel, utilisée par les instructions LOF, LOC, EOF, FPOS, etc du Disk-Basic

Rôle : Utilisé par la Disk-ROM.

0FE99h ***H.LOC***

Appel : Au début de la routine LOC (« LOCation »), utilisée par l'instruction LOC du Disk-Basic

Rôle : Utilisé par la Disk-ROM.

0FE9Eh ***H.LOF***

Appel : Au début de la routine LOF (« Length Of File »), utilisée par l'instruction LOF du Disk-Basic

Rôle : Utilisé par la Disk-ROM.

0FEA3h ***H.EOF***

Appel : Au début de la routine EOF (« End Of File»), utilisée lors de l'instruction EOF du Disk-Basic.

Rôle : Utilisé par la Disk-ROM.

0FEA8h ***H.FPOS***

Appel : Au début de la routine FPOS (« File POSition »).

Rôle : Utilisé par la Disk-ROM.

0FEADh ***H.BAKU***

Appel : Au début de la routine BAKUPT (« BAKe UP »).

Rôle : Utilisé par la Disk-ROM.

0FEB2h ***H.PARD***

Appel : Au début de la routine PARDEV (« PARse DEViCe name »).

Rôle : Permet de changer le nom du device « logical device name ».

0FEB7h	<i>H.NODE</i>
Appel :	Au début de la routine NODEVN (« NO DEViCe Name »), pas de nom trouvé dans la table des DEVICE.
Rôle :	Permet de mettre un nom au disque dont le nom a été omis.
0FEBCh	<i>H.POSD</i>
Appel :	Au début de la routine POSDSK (« POSsible DiSK »)
Rôle :	Permet de raccorder un disque.
0FEC1h	<i>H.DEVN</i>
Appel :	Au début de la routine DEVNAM (« DEViCe NAME »).
Rôle :	Permet d'étendre un nom de disque logique.
0FEC6h	<i>H.GEND</i>
Appel :	Au début de la routine GENDSP (« GENERal device DISPatcher »), traitement d'un nom de disque logique.
Rôle :	Permet d'étendre un nom de disque logique.
0FECBh	<i>H.RUNC</i>
Appel :	Au début de la routine RUNC (« RUN Clear »), utilisée lors des instructions NEW et RUN du Basic.
Rôle :	Permet le détournement des instructions NEW et RUN.
0FED0h	<i>H.CLEA</i>
Appel :	Au début de la routine CLEARC (« CLEAR Clear »), utilisée lors de la remise à zéro des variables Basic.
Rôle :	Permet d'éviter l'effacement des variables Basic.
0FED5h	<i>H.LOPD</i>
Appel :	Au début de la routine LOPDFT (« LOoP and set DeFault »), initialisation de la table des variables.
Rôle :	Permet de détourner la routine.
0FEDAh	<i>H.STKE</i>
Appel :	Au début de la routine STKERR (« STAcK ERRor »), utilisée lors des instructions Basic FOR et GOSUB.
Rôle :	lors de l'initialisation du Disk-Basic ou du MSX-DOS, le MSX vérifie si ce Hook a été modifié par une éventuelle cartouche utilisant le lecteur de disquette (contenu de 0FEDAh différent de 0C9h). Si tel est le cas, le MSX saute au Hook (JP 0FEDAh) lorsque l'installation du DOS est terminée.

0FEDFh	<i>H.ISFL</i>
Appel :	Au début de la routine ISFLIO (« IS FiLe I/O »), test d'existence d'un fichier
Rôle :	Permet de détourner la routine.
0FEE4h	<i>H.OUTD</i>
Appel :	Au début de la routine OUTDO (« OUT DO »), sortie d'un caractère sur écran ou imprimante
Rôle :	Permet de détourner la routine.
0FEE9h	<i>H.CRDO</i>
Appel :	Au début de la routine CRDO (« CRIf DO »), émission d'un CR (carriage return - code 0Dh) puis d'un LF (line feed - code 0Ah).
Rôle :	Permet, par exemple, d'adapter les sorties sur une imprimante possédant un line-feed automatique.
0FEEeh	<i>H.DSKC</i>
Appel :	Au début de la routine DSKCHI (« DiSK CHAracter Input »)
Rôle :	Permet de détourner la routine.
0FEF3h	<i>H.DOGR</i>
Appel :	Au début de la routine DOGRPH (00085h en Sub-ROM), utilisée par les fonctions graphiques du Basic.
Rôle :	Permet de détourner la routine.
0FEF8h	<i>H.PRGE</i>
Appel :	Au début de la routine PRGEND (« PRoGram END »), appelée à la fin de l'exécution d'un programme Basic.
Rôle :	Permet n'importe quelle action avant de rendre la main à l'utilisateur.
0FEFDh	<i>H.ERRP</i>
Appel :	Au début de la routine ERRPRT (« ERRor PRinT »), utilisée lors de l'affichage par le Basic d'un message d'erreur.
Rôle :	Permet de traiter les erreurs directement.
0FF02h	<i>H.ERRF</i>
Appel :	à la fin de la routine affichant un message d'erreur sous Basic.
Rôle :	Permet une action supplémentaire après une erreur.

0FF07h	<i>H.READ</i>
Appel :	Au début de la routine READY, utilisée à chaque fois que le message « Ok » (ou le prompt défini par l'utilisateur sur MSX2) est affiché à l'écran.
Rôle :	Permet de faire beaucoup de choses, par exemple provoquer l'émission d'un bref bip sonore à chaque affichage de « Ok ».
0FF0Ch	<i>H.MAIN</i>
Appel :	Au début de la routine MAIN (« MAIN entry »), utilisée à chaque accès à l'interpréteur Basic (autrement dit : en permanence sous Basic).
Rôle :	Permet de programmer une interruption par exemple.
0FF11h	<i>H.DIRD</i>
Appel :	Au début de la routine DIRDO (« DIRect DO »), utilisé lors de l'exécution d'une instruction en mode direct.
Rôle :	Permet toutes les manipulations sur le mode direct du Basic.
0FF16h	<i>H.FINI</i>
Appel :	Au début de la routine FININT (« Function INterpretation INiTialize »), utilisée lors de la traduction d'une instruction Basic par l'interpréteur.
Rôle :	Permet de détourner le traitement des instructions Basic.
0FF1Bh	<i>H.FINE</i>
Appel :	Au début de la routine FINEND (« Function INterpretation END »), utilisée à la fin de l'interprétation d'une instruction Basic.
Rôle :	Permet de détourner la routine.
0FF20h	<i>H.CRUN</i>
Appel :	Au début de la routine CRUNCH, transformation d'une ligne Basic en mots-clefs.
Rôle :	Permet de détourner la routine.
0FF25h	<i>H.CRUS</i>
Appel :	Au début de la routine CRUSH, recherche d'un mot-clef dans la liste alphabétique.
Rôle :	Permet de détourner la routine.
0FF2Ah	<i>H.ISRE</i>
Appel :	Au début de la routine ISRESV (« IS token RESerVed »), découverte d'un mot-clef lors de l'exécution de la routine CRUNCH.
Rôle :	Permet de détourner la routine.

0FF2Fh	<i>H.NTFN</i>
Appel :	Au début de la routine NTFN2, utilisée lorsqu'un mot-clef est suivi par un numéro de ligne.
Rôle :	Permet de détourner la routine.
0FF34h	<i>H.NOTR</i>
Appel :	Au début de la routine NOTRSV (« token NOT ReSerVed »), la suite de caractères examinée ne constitue pas un mot-clef.
Rôle :	Permet de détourner la routine.
0FF39h	<i>H.SNGF</i>
Appel :	Au début de la routine SNGFOR, accès aux fonctions mathématiques BCD
Rôle :	Permet d'installer de nouvelles routines mathématiques.
0FF3Eh	<i>H.NEWS</i>
Appel :	Au début de la routine NEWSTT (« NEW STaTement »), passage à une nouvelle instruction Basic.
Rôle :	Permet de détourner la routine.
0FF43h	<i>H.GONE</i>
Appel :	Au début de la routine GONE2, instruction de déroutement en cours
Rôle :	Permet de détourner la routine.
0FF48h	<i>H.CHRG</i>
Appel :	Au début de la routine CHRGET (« CHaRacter GET »), saisie d'un caractère au clavier.
Rôle :	Permet de détourner la routine.
0FF4Dh	<i>H.RETU</i>
Appel :	Au début de la routine RETURN (« RETURN from subroutine ») utilisée lors de l'instruction Basic RETURN.
Rôle :	Permet de détourner la routine.
0FF52h	<i>H.PRTF</i>
Appel :	Au début de la routine PRTFLD, utilisée lors de l'instruction Basic PRINT
Rôle :	Permet de détourner la routine.
0FF57h	<i>H.COMP</i>
Appel :	Au début de la routine COMPRT, boucle interne à l'instruction Basic PRINT

Rôle : Permet de détourner la routine.

0FF5Ch ***H.FINP***

Appel : Au début de la routine FINPRT (« FINal PRinT »), fin de l'instruction Basic PRINT.

Rôle : Permet de détourner la routine.

0FF61h ***H.TRMN***

Appel : Au début de la routine TRMNOK, traitement d'une donnée incorrecte lors des instructions Basic DATA et INPUT.

Rôle : Permet de détourner la routine.

0FF66h ***H.FRME***

Appel : Au début de la routine interne FRMEVL (« FoRMula EVaLuator ») de reconnaissance d'une expression dans le texte d'un programme Basic.

Rôle : Permet d'installer de nouvelles routines mathématiques.

Note : Voici la description de la routine FRMEVL.

Entrée : HL = Pointeur du texte.

Sortie : HL = Pointeur vers l'expression trouvée.

VALTYP (F663h) = Type de valeur de l'expression.

DAC (F7F6h) = Valeur trouvée.

0FF6Bh ***H.NTPL***

Appel : Au début de la routine utilisée lors de FRMEVL.

Rôle : Permet d'installer de nouvelles routines mathématiques.

0FF70h ***H.EVAL***

Appel : Au début de la routine EVALST (« EVALuate SStatement »), évaluation d'une expression.

Rôle : Permet de détourner la routine.

0FF75h ***H.OKNO*** ou ***H.MDIN***

Appel : Au début de la routine de calcul de fonctions transcendentes (H.OKONO) ou au début de la routine d'interruption de l'entrée du MSX-Midi (H.MDIN).

Rôle : Permet de détourner la routine ou, de gérer les interruptions de l'entrée Midi.

0FF7Ah ***H.FING***

Appel : A la fin de la routine de calcul de fonctions transcendentes.

Rôle : Permet de détourner la routine.

0FF7Fh	<i>H.ISMI</i>
Appel :	Au début de la routine ISMID\$ (« IS MID\$ »), utilisée lors de l'instruction MID\$ du Basic.
Rôle :	Permet de détourner la routine.
0FF84h	<i>H.WIDT</i>
Appel :	Au début de la routine WIDTHS (« WIDTH of Screen »), utilisée lors de l'instruction Basic WIDTH.
Rôle :	Permet de détourner la routine.
0FF89h	<i>H.LIST</i>
Appel :	Au début de la routine LIST , utilisée lors des instructions Basic LIST et LLIST.
Rôle :	Permet de détourner la routine.
0FF8Eh	<i>H.BUFL</i>
Appel :	Au début de la routine BUFLIN (« Buffer LINE »), utilisée lors de l'instruction Basic LIST.
Rôle :	Permet de détourner la routine.
0FF93h	<i>H.FRQI</i> ou <i>H.MDTM</i>
Appel :	Au début de la routine FRQINT utilisée lors de l'instruction Basic POKE (H.FRQI) ou au début de la routine d'interruptions HMDTM du timer du MSX-Midi (H.MDTM).
Rôle :	Permet de détourner la routine ou, de gérer l'interruption du timer Midi qui se produit toutes les 5 ms.
0FF98h	<i>H.SCNE</i>
Appel :	Au début de la routine SCNEX2, conversion d'un numéro de ligne en adresse mémoire et inversement.
Rôle :	Permet de détourner la routine.
0FF9Dh	<i>H.FRET</i>
Appel :	Au début de la routine FRETMP (« FREe up TeMPories »), recherche d'un emplacement libre pour stocker une variable alphanumérique
Rôle :	Permet de détourner la routine.
0FFA2h	<i>H.PTRG</i>
Appel :	Au début de la routine PTRGET (« PoinTeR GET »).

Rôle : Permet d'utiliser d'autres noms de variables.

0FFA7h ***H.PHYD***

Appel : Au début de la routine PHYDIO (« PHYsical Disk I/O ») du Bios, adresse 00144h en Main-ROM.

Rôle : Utilisé par la Disk-ROM, Permet de détourner la routine.

0FFACh ***H.FORM***

Appel : Au début de la routine FORMAT (« disk FORMATter ») du Bios, adresse 00147h en Main-ROM.

Rôle : Utilisé par la Disk-ROM.

0FFB1h ***H.ERRO***

Appel : Au début de la routine ERROR de traitement d'une erreur sous Basic. Le registre A contient le numéro d'erreur (voir le [tableau des codes d'erreurs du Basic](#)).

Rôle : Permet de traiter les erreurs avec sa propre routine.

0FFB6h ***H.LPTO***

Appel : Au début de la routine LPTOUT (« Line PrinTer OUTput »), sortie d'un caractère sur imprimante.

Rôle : Permet d'utiliser une imprimante non-MSX.

0FFBBh ***H.LPTS***

Appel : Au début de la routine LPTSTT (« Line PrinTer STaTus »), test de l'état de l'imprimante

Rôle : Permet d'utiliser une imprimante non-MSX

0FFC0h ***H.SCRE***

Appel : Appelée au début d'une routine de changement d'écran, utilisée par l'instruction Basic SCREEN.

Rôle : Permet de détourner la routine.

0FFC5h ***H.PLAY***

Appel : Appelée au début de la routine PLAY, utilisée par l'instruction Basic PLAY.

Rôle : Permet de détourner la routine.

6 Le processeur vidéo (VDP)

6.1 Avertissement

Avec le processeur vidéo, nous abordons un domaine plus spécifique aux MSX. Ce livre contient tous les renseignements nécessaires pour faire fonctionner correctement le processeur vidéo de chaque version de MSX en ne tenant compte que de chaque mode d'écran du MSX1 à MSX turbo R. Quant aux registres, ils seront entièrement détaillés afin de pouvoir y accéder directement sans avoir de passer par le BIOS.

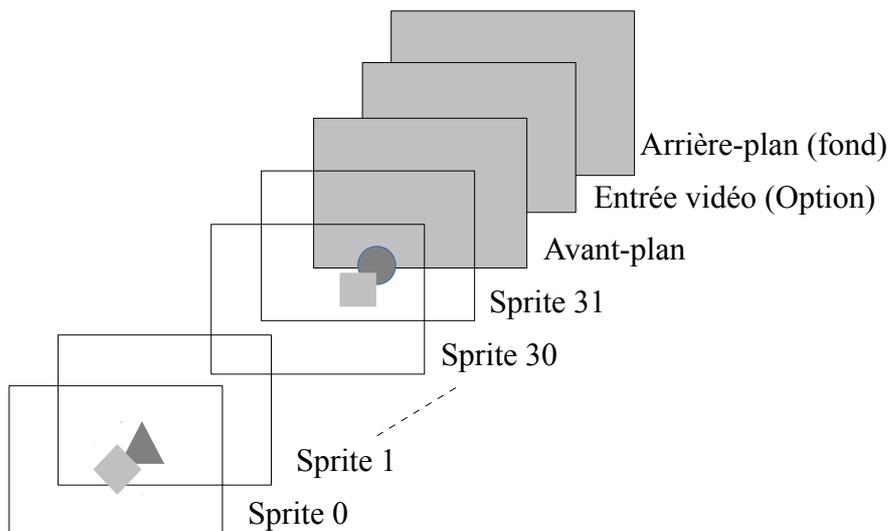
6.2 Introduction au processeur vidéo

Le processeur vidéo ou VDP (« Video Display Processor ») est sans aucun doute l'élément le plus excitant du système MSX. Surtout sur MSX2 avec sa mémoire vidéo énorme (128 Ko en général) qui offre une résolution graphique extraordinaire pour du matériel grand public de l'époque. Il autorise l'affichage de deux cent cinquante six couleurs simultanément à l'écran sur MSX2 et plusieurs milliers sur MSX2+. Le VDP comporte par ailleurs un jeu complet de commandes graphiques permettant notamment le tracé de lignes, de rectangles pleins ou vides, de scrolling pour ne citer que quelques fonctions. La puissance du V99x8 vient du fait que c'est un processeur VLSI (Very Large Scale Integration), c'est-à-dire qu'il atteint un degré d'intégration encore inégalé en micro-informatique. Le V9938 restera dans l'histoire comme le premier processeur VLSI présent en informatique grand public.

6.3 Fonctionnement du VDP

Le principe d'un processeur vidéo est relativement simple. Il s'agit de soulager le micro-processeur central de toutes les tâches liées au graphisme en lui adjoignant un second processeur dédié à l'affichage. Au niveau de la programmation, on peut ignorer totalement la présence du processeur et n'utiliser que les routines du Bios (tracé de lignes, chargement de pages, Sprites, etc). Cette solution, si elle offre l'avantage de la simplicité, prive le programmeur de toute une série de fonctions non-disponibles avec le Bios (scrolling, clignotement pour ne citer que deux exemples). Il est donc intéressant de pouvoir accéder directement au processeur vidéo. Ceci s'opère par l'intermédiaire de registres, huit sur MSX1 et quarante sept sur MSX2 - ce qui vous donne une idée de la différence de puissance entre le processeur du MSX1 et celui du MSX2. La plupart des registres définissent des paramètres précis, la couleur du texte par exemple, alors que quelques registres ont une fonction spéciale, écrire dans le registre 46 déclenche automatiquement une opération, le tracé d'une ligne par exemple. Vous trouverez tout au long de ce chapitre l'explication du contenu de chaque registre ainsi qu'un exposé des différents modes graphiques et des Sprites.

Avant de voir tout ça, voici un petit schéma qui montre les priorités d'affichage du VDP pour se donner une image de ce quoi on parle par la suite.



6.4 Comment accéder au VDP

Il existe deux moyens d'accéder au processeur vidéo. Tout dépend de ce que vous cherchez à faire et du MSX utilisé.

1. Soit vous appelez simplement les routines du Bios en Main-ROM ou en Sub-ROM.
2. Soit vous décidez d'y accéder directement pour une application qui nécessite une vitesse la plus élevée possible. L'utilisation du Bios ralentit légèrement les accès au VDP.

Dans le premier cas, la solution est évidente : utilisez les routines WRTVDP, RDVDP, SETPLT, GETPLT et VDPSTA du Bios.

Dans le second cas, la situation est moins simple, sans passer par le Bios, point de salut, vous risquez de perdre la compatibilité dès que vous utilisez l'instruction OUT ou IN du Z80. Heureusement, Microsoft et ASCII ont prévu ce cas de figure lors de la conception du système MSX. Voici la marche à suivre pour accéder directement au VDP par les ports E/S :

Dans le BIOS en Main-ROM, il y a 2 octets qui servent à indiquer l'adresse du port d'E/S du VDP. Le premier port de lecture est indiqué par ce que contient la ROM à l'adresse 0006h et le premier port d'écriture par le contenu de l'adresse suivante. Les autres ports se trouve à la suite comme indiqué dans le tableau suivant.

Ports de lecture	Adresse du sortie	Fonction
0	Contenu de VDP.DR (0006h en Main-ROM)	Lire en VRAM
1	Contenu de VDP.DR (0006h en Main-ROM)+1	Lecture du registre de statut
Ports d'écriture	Adresse du port d'entrée	Fonction

0	Contenu de VDP.DW (0007h en Main-ROM)	Écrire en VRAM
1	Contenu de VDP.DW (0007h en Main-ROM) +1	Écrire dans les registres
2	Contenu de VDP.DW (0007h en Main-ROM) +2	Écrire dans la palette de
3	Contenu de VDP.DW (0007h en Main-ROM) +3	Envoi de données successives dans un registre de contrôle ou une suite de registres

Les MSX1 sont équipés en général d'un VDP de première génération (TMS9918 à TMS9929 selon la sortie). Ces VDP ont deux ports d'écriture et deux de lecture (ports 0 et 1).

Les MSX de générations suivantes sont équipés en général d'un VDP V9938 (MSX2) ou d'un V9958 (MSX2+ et MSX turbo R). Ces VDP ont deux ports d'écriture (ports 3 et 4) supplémentaires.

Notes : - Certains MSX1 sont équipés d'un V9938 (Yamaha CX5MII et Spectravideo SVI-738). Il y a même certains MSX2 équipés d'un V9958. Leur Bios ne contiennent cependant que les routines correspondantes à leur génération.

- Dans les faits, tous les MSX utilisent les ports de sortie 098h et 099h et les ports d'entrée 098h à 09Bh pour accéder au VDP interne. Seuls les VDP externes utilisent des ports différents.

Ecrire dans un registre de contrôle (0 ~ 23, 25 et 32 ~ 46)

1. Lire le contenu de la case mémoire à l'adresse 00007h en Main-ROM. Vous aurez ainsi l'adresse d'accès au port 0 du VDP. L'adresse du port 1 sera la suivante. (Si vous ne comprenez pas la signification de Main-ROM, voyez le chapitre concernant les Slots.)
1. Envoyer la donnée à écrire sur le port 1. (« OUT »).
2. Envoyer, toujours sur le port 1, le numéro du registre dans lequel vous voulez écrire en gardant le bit de poids fort à 1, ce qui revient à ajouter 80h au numéro du registre.

Un exemple :

< Mettre le bit 2, du registre de contrôle 9 du VDP, à 0. >

```

ORG 0C000H
;
; Le programme suivant active le mode d'affichage
; à 60 Hertz. Sur un téléviseur ou un moniteur
; en 50 Hz, l'image "sautera". Il faut remettre
; le bit 2 à 0 dans le registre neuf afin de
; rétablir une image normale.
;
DEBUT: LD A, (7) ; C = Numéro du port E/S correspondant
LD C, A ; au port 0 d'écriture du VDP.
INC C ; Port E/S du port 1 d'écriture.
;
LD A, (0FFE8h) ; Met le contenu de RG09SAV dans A
AND 0FDH ; Met le bit 2 à 0
DI ; Interruptions interdites
OUT (C), A ; Envoi sur le port 1
LD (0FFE8h), A ; Actualise RG09SAV
LD A, 80H+9 ; Numéro de registre avec bit 7 à 1
OUT (C), A ; Envoi sur le port 1

```

```

        EI                ; Interruptions autorisées
        RET
;
; NOTE : Ce programme lit le contenu de RG09SAV afin de ne pas
; modifier les autres bits du registre
;
        END DEBUT

```

L'équivalent en Basic de ce programme serait :

```
C=PEEK(7):C=C+1:OUT C,(PEEK(&HFFE8)AND &HFD):OUT C,&H80+9
```

ou mieux :

```
VDP(10)=VDP(10)AND &HFD
```

Dans certains cas, il peut être utile d'écrire dans une suite de registres ou d'écrire plusieurs fois dans le même registre. Pour cela, l'accès indirect est possible grâce au port 3 du processeur vidéo :

1. Écrire le numéro du registre dans lequel vous désirez écrire dans le registre 17 de contrôle avec la méthode précédente.
1. Envoyez vos données, les unes à la suite des autres sur le port 3 du VDP.

L'opération d'auto-incrémentation a normalement lieu, à savoir que la valeur dans le registre 17 augmente à chaque accès au port 3. Il est possible d'annuler cette fonction de manière à toujours écrire dans le même registre. Il suffit de mettre à 1 le bit 7 du registre 17, ce qui revient à ajouter 80h.

Exemple d'écriture indirecte :

< Mettre le bit 2 du registre de contrôle 9 à 0, attendre un peu puis, remise du bit 2 à 1. >

```

        ORG 0C000H
DEBUT:  LD A,(7)           ; C = Numéro du port E/S correspondant
        LD C,A           ; au port 0 d'écriture du VDP.
        INC C           ; Port E/S du port 1 d'écriture.
;
        LD A,80H+9      ; Registre 9 sans auto-incrémentation...
        DI             ; Interruptions interdites
        OUT (C),A
        LD A,80H+17     ; ...dans registre 17 (+80h)
        OUT (C),A
        EI             ; Interruptions autorisées
;
        INC C           ; Port E/S du port 2 dans C
        INC C           ; Port E/S du port 3 dans C
        LD A,(0FFE8h)   ; Contenu de RG09SAV dans A
        AND 0FDH        ; Mettre le bit 2 à 0 (mode 60 Hz)
        OUT (C),A      ; Envoi sur le port 3
        LD (0FFE8h),A   ; Actualise RG09SAV
;
        CALL ATTEND     ; Appel le sous-programme d'attente
;
        OR 2            ; Met le bit 2 à 1 (mode 50 Hz)
        LD (0FFE8h),A   ; Actualise RG09SAV
        OUT (C),A      ; Envoi sur le port 3

```

```

RET
;
ATTEND: LD B,A ; Préserve A
        PUSH BC ; et BC
        LD BC,0 ; Boucle d'attente
BCL:    DEC BC ;
        NOP ;
        LD A,B ; pour voir un peu
        OR C ;
        CP 0 ;
        JR NZ,BCL ; quelque chose
        POP BC ; Restitue BC
        LD A,B ; et A
        RET

```

Ecrire dans un registre de la palette des couleurs

1. Écrire le numéro de la couleur à modifier (0 ~ 15) dans le registre de contrôle 16.
1. Envoyez sur le port 2 du VDP, deux octets qui codent la nouvelle teinte au format suivant.

Premier octet :

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	rouge (0~7)			0	bleu (0~7)		

Deuxième octet :

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	0	0	0	0	vert (0~7)		

Exemple :

< Changer la couleur n°1 (noir), en vert. >

```

ORG 0C000H
DEBUT: LD A,(7) ; C = Numéro du port E/S correspondant
        LD C,A ; au port 0 d'écriture du VDP.
        INC C ; Port E/S du port 1 d'écriture.
;
        LD A,1 ; A=1
        DI ; Interruptions interdites.
        OUT (C),A ; Couleur 1 dans
        LD A,80H+16 ; le
        OUT (C),A ; registre 16.
        EI ; Interruptions autorisées
;
        INC C ; Port E/S du port 2 d'écriture
        LD A,42H ; rouge=4 bleu =2
        OUT (C),A ; et hop...
        LD A,5 ; vert=5
        OUT (C),A ; et re-hop...
;
        RET
END DEBUT

```

Lire un registre de statut du VDP

1. Écrire le numéro de registre de statut que vous désirez lire (0 ~ 9) dans le registre de contrôle 15.
1. Lire le port 1 du VDP pour obtenir le contenu du registre. (Sur MSX1, seule cette étape est nécessaire.)
2. Remettre 0 dans le registre de contrôle 15. Cette étape est nécessaire à partir du MSX2 car à chaque interruption le Bios lit le registre de statut 0 sans tenir compte du registre de contrôle 15.

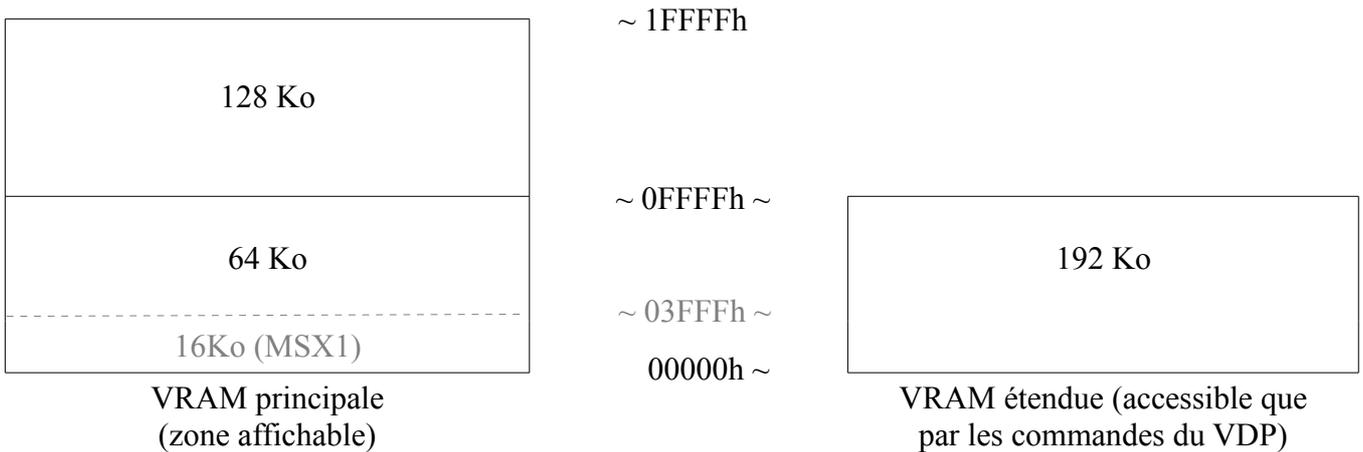
Exemple : < lire le registre de statut 1 >

```
                ORG 0C000H
;
DEBUT:         LD A,(7)           ; C = Numéro du port E/S correspondant
                LD C,A           ; au port 0 d'écriture du VDP.
                INC C            ; Port E/S du port 1 d'écriture.
                LD B,C           ; Préserve le Numéro de port dans B
;
                LD A,0           ;
                DI               ; Interruptions interdites
                OUT (C),A        ; Numéro du registre de statut...
                LD A,80H+15      ;
                OUT (C),A        ; ...dans le registre 15
;
                LD A,(6)         ; C = Numéro du port E/S correspondant
                LD C,A           ; au port 0 de lecture du VDP
                INC C            ; Port E/S du port 1 d'écriture
                IN (C),A         ; Lecture du registre
;
                LD B,C           ; Restitue le Numéro de port 1
                LD A,0           ;
                OUT (C),A        ; Registre de statut 0
                LD A,80H+15      ;
                OUT (C),A        ; dans le registre 15
                EI               ; Interruptions autorisées
;
                RET
```

6.5 Lecture et écriture dans la mémoire vidéo

Un MSX peut être équipé de 16, 64, 128 ou 192 Ko de mémoire vive réservée à la vidéo (VRAM). Tous les MSX2 commercialisés en France comportent 128 Ko. Cette mémoire ne se trouve pas dans un Slot, ce faisant, le Z80 ne peut en aucun cas l'adresser, seul le processeur vidéo peut le faire.

Carte de la VRAM



Il existe trois méthodes permettant de manipuler la mémoire vidéo. La première consiste à utiliser le Bios, puisqu'on y trouve les routines WRTVRM, RDVRM qui respectivement écrivent et lisent la mémoire vidéo. Les deux autres méthodes nécessitent des accès directs au processeur vidéo, l'une passe par l'envoi de l'adresse à laquelle on désire accéder puis lecture/écriture de la donnée, alors que l'autre passe par les commandes du processeur vidéo. Pour ce qui est de cette dernière solution, elle est détaillée dans la partie concernant les commandes propres au VDP.

Nous allons à présent détailler en 5 étapes la méthode d'accès à la VRAM par envoi de l'adresse afin de lire ou écrire une donnée.

1. Choisir la VRAM principale ou la VRAM secondaire en agissant sur le bit 6 du registre 45 du VDP (0=VRAM principale, 1=VRAM étendue). Cette opération n'est nécessaire que lors du premier accès vidéo. En général, vous n'avez pas à vous occuper de cette étape. Le MSX ayant 192 Ko de VRAM sont rares.
2. Sur MSX2 ou plus récent, l'adressage de la VRAM est codée sur 17 bits (0 à 1FFFFh). Le registre 14 sert à manipuler les 3 bits de poids fort de l'adresse qui vous intéresse. Charger ces 3 bits (A16, A15 et A14) dans le registre 14. (Sur MSX1, vous n'avez pas à vous occuper de cette étape.)
3. Envoyer sur le port 1 du VDP les 8 bits de poids faible de l'adresse (bits A7 à A0).
4. Envoyer à nouveau sur le port 1 les bits manquants de l'adresse (bits A13 à A8) ainsi que l'instruction d'écriture ou de lecture en VRAM.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Octet :	00=lecture, 01=écriture	A13	A12	A11	A10	A9	A8	

5. Lire ou écrire la donnée sur le port 0 du VDP. L'auto-incrémentation de l'adresse ayant lieu, il n'est pas nécessaire de redéfinir l'adresse pour accéder à chaque octet suivant.

Exemple :

< en SCREEN 0 / 40 colonnes, envoyer des caractères dans la table des caractères afin d'afficher un message en haut à droite de l'écran >

```
                ORG 0C000H
;
DEBUT:         LD A,(7)           ; C = Numéro du port E/S correspondant
                LD C,A           ; au port 0 d'écriture du VDP.
                INC C            ; Port E/S du port 1 d'écriture.
;
                LD A,0           ; 3 bits de poids fort
                DI               ; Interruptions interdites
                OUT (C),A        ; 3 bits de poids fort
                LD A,80H+14      ; 3 bits de poids fort
                OUT (C),A        ; dans le registre 14
;
                LD A,01FH        ; 8 bits de
                OUT (C),A        ; poids faible = 31
;
                LD A,040H        ; Bits 6~7 mis à 01 pour une
                OUT (C),A        ; écriture + autres bits de l'adresse
                EI               ; Interruptions autorisées
;
                EX (SP),HL       ; Temps
                EX (SP),HL       ; d'attente pour VDP MSX1
;
                DEC C            ; Numéro du port E/S du port 0 dans C
                LD HL,DATA       ; Position des données
                LD B,9           ; Neuf lettres
                OTIR             ; Transfert des données sur le port 0
                RET
DATA:          DB 'CA MARCHE'
                END DEBUT
```

6.6 Les registres de contrôle du processeur video.

Ces registres permettent de modifier le comportement du VDP. Ils ne sont accessibles qu'en écriture. Il est donc conseillé, voir nécessaire, d'actualiser les variables système juste après chaque écriture à un registre de 0 à 25 par accès direct.

Voici la liste complète de tous les registres de contrôle et la signification des informations qu'ils contiennent :

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Registre 0 :	0	DG	IE2	IE1	M5	M4	M3	EV

EV = Entrée vidéo externe. 1 pour l'activer ; 0 pour la désactiver (valeur par défaut).

M3 = Bit de mode graphique.

M4 = Bit de mode graphique. (Inutilisé sur MSX1.)

M5 = Bit de mode graphique. (Inutilisé sur MSX1.)

IE1 = Autorisation des interruptions qui se produisent juste avant le balayage de la ligne horizontale indiquée au registre 19. (Inutilisé sur MSX1.)

IE2 = Autorisation des interruptions du crayon optique. (Inutilisé sur MSX, mettre à 0.)
(N'existe pas sur le V9958)

DG = 1 pour mettre le bus de couleur en mode « entrée » afin de récupérer les données en VRAM sur les MSX équipé d'un digitaliseur. (Inutilisé sur MSX1.)

Note : Pour plus de détails sur les bits de mode graphique, voir le registre 1 du VDP ci-dessous.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
4/16k	BL	IE0	M1	M2	0	SI	MAG

MAG = Doublement de la taille des Sprites : 1 = Double taille ; 0 = Taille ordinaire.

SI = Taille des Sprites : 1 pour 16x16 ; 0 pour 8x8.

M2 = Bit de mode graphique.

M1 = Bit de mode graphique.

IE0 = Autorise l'interruption qui se produit à chaque fin d'affichage de l'écran.

BL = Affichage de l'écran. 1 pour activer (valeur par défaut) ; 0 pour désactiver.

4/16k = Configuration de la VRAM. 1 = 16Ko ; 0 = 4K. (MSX1 seulement)

Note : Le mode graphique est déterminé par les bits M1 à M5 de la manière suivante.

	M5	M4	M3	M2	M1	
SCREEN 0	0	0	0	0	1	40 colonnes, 2 couleurs (4 couleurs possible sur MSX2~)
SCREEN 0	0	1	0	0	1	80 colonnes, 2 couleurs (4 couleurs possible)
SCREEN 1	0	0	0	0	0	32 colonnes, 16 couleurs (2 couleurs par caractères)
SCREEN 2	0	0	1	0	0	256x192, 16 couleurs (2 couleurs chaque 8 pixels)
SCREEN 3	0	0	0	1	0	64x48, 16 couleurs (sans contrainte)
SCREEN 4	0	1	0	0	0	Equivalent au SCREEN 2 mais avec Sprites MSX2
SCREEN 5	0	1	1	0	0	256x212, 16 couleurs (sans contrainte)
SCREEN 6	1	0	0	0	0	512x212, 4 couleurs (sans contrainte)
SCREEN 7	1	0	1	0	0	512x212, 16 couleurs (sans contrainte)
SCREEN 8	1	1	1	0	0	256x212, 256 couleurs (sans contrainte)
SCREEN 9	Idem au SCREEN 6					40 colonnes en Hangeul, 4 couleurs (MSX2 coréen seulement)
SCREEN 10	Idem au SCREEN 8					256x212, 12599 couleurs (avec contrainte)
SCREEN 11	Idem au SCREEN 8					Idem au SCREEN 10
SCREEN 12	Idem au SCREEN 8					256x212, 19268 couleurs (avec contrainte)

- Le mode SCREEN 9 est en fait le même mode que le SCREEN 6 mais géré à la façon d'un mode texte par le Basic. Ce mode n'existe que sur les MSX2 coréens.
- Le mode SCREEN 10 et 11 est obtenu en mettant les bits YJK et YAE du registre 25 à 1. La différence entre les deux se situe dans la gestion des couleurs sous Basic (RGB ou YJK).
- Le mode SCREEN 12 est obtenu en mettant les bits YJK à 1 et YAE à 0 (registre 25).

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	N16	N15	N14	N13	N12	N11	N10

Registre 2 :

Ce registre détermine l'emplacement de la table qui constitue l'avant-plan.

Modes d'écran MSX1 :

N10 à N13 = Ces bits correspondent aux quatre bits de poids fort de l'adresse du début de la table des caractères en VRAM. L'adresse véritable s'obtient en multipliant la valeur de ces 4 bits par 400h. Par exemple, si les bits sont 1001 (soit 9 en hexa), la table des caractères se trouve en $9 \times 400h = 2400h$.

N14 à N16 = N'existent pas. L'adresse peut varier dans ces conditions entre 0 et 3C00h.

Autres modes d'écran :

N10 à N14 = Ces bits doivent toujours être à 1.

N15 et N16 = Ces bits déterminent le numéro de page (Bits de poids fort de la table bitmap).

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
C13	C12	C11	C10	C9	C8	C7	C6

Registre 3 :

Modes d'écran MSX1 :

C6 à C13 = Ces bits codent les 8 bits de poids fort de l'adresse du début de la table des couleurs. L'adresse réelle en mémoire s'obtient donc en multipliant le contenu du registre 3 par 40h. Par exemple si l'on lit 0B6h dans le registre 3, l'adresse de début de la table des couleurs est $0B6h \times 40h = 2D80h$. L'adresse peut varier entre 0 et 3FC0h.

Attention : En SCREEN 2, le fonctionnement est différent, la table des couleurs ne peut se trouver qu'en 0 ou 2000h. Seul le bit de poids fort intervient. Les autres bits sont tous à 1. Le registre 3 ne devra donc contenir que 7Fh ou FFh.

Autres modes d'écran :

C6 à C13 = Ces bits s'utilisent avec C14, C15 et C16 du registre 10 du VDP. Tous ces 11 bits codent l'adresse du début de la table des couleurs comme sur MSX1 (voir ci-dessus). L'adresse peut varier entre 0 et 1FFC0h.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	0	F16	F15	F14	F13	F12	F11

Registre 4 :

Modes d'écran MSX1 :

F11 à F13 = Ces bits codent les 3 bits de poids fort de l'adresse du début de la table des formes en VRAM. L'adresse véritable s'obtient donc en multipliant la valeur de ces 3 bits par 800h. Par exemple, si les bits sont 100 (soit 4 en hexa), la table des formes se trouve en $4 * 800h = 2000h$.

F14 à F16 = N'existent pas. L'adresse peut varier dans ces conditions entre 0 et 3800h.

Attention : En SCREEN 2, le fonctionnement est différent. La table des formes ne peut commencer qu'en 0 ou 2000h. Le registre ne peut avoir que 03h ou 07h comme valeur. De surcroît, F14 doit toujours être positionné à l'inverse du bit 8 du registre 3. (Vous me suivez bien ?)

Autres modes d'écran :

F11 à F16 = Ces bits codent les 6 bits de poids fort de l'adresse du début de la table des formes. Le fonctionnement est le même que sur MSX1 (voir ci-dessus). L'adresse varie entre 0 et 1F800h.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
S14	S13	S12	S11	S10	S9	S8	S7

Registre 5 :

Modes d'écran MSX1 :

S7 à S13 = Ces bits codent les 7 bits de poids fort de l'adresse du début de la table des attributs de Sprites. L'adresse réelle s'obtient donc en multipliant le contenu du registre 5 par 80h. Par exemple si l'on lit 037h dans le registre 5, l'adresse du début de la table des attributs de Sprites est $037h * 80h = 1B80h$.

S14 = Ignoré. L'adresse peut varier entre 0 et 3F80h.

Autres modes d'écran :

S7 à S14 = S'utilisent avec S15 et S16 du registre 11 du VDP. Ces 10 bits codent l'adresse du début de la table des attributs de Sprites comme sur MSX1 (voir ci-dessus). L'adresse peut varier entre 0 et 1FF80h.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Registre 6 :	0	0	P16	P15	P14	P13	P12	P11

Modes d'écran MSX1 :

P11 à P13 = Codent les 3 bits de poids fort de l'adresse du début de la table de génération des Sprites en VRAM. L'adresse véritable s'obtient donc en multipliant la valeur de ces 3 bits par 800h. Par exemple, si les bits sont 100 (soit 4 en hexa), la table des formes de Sprites se trouve en 04h*800h=2000h.

P14 à P16 = Ignorés. L'adresse peut varier dans ces conditions entre 0 et 3800h.

Autres modes d'écran :

P11 à P16 = Codent les 6 bits de poids fort e l'adresse du début de la table de génération des Sprites. Le fonctionnement est le même que sur MSX1 (voir ci-dessus). L'adresse varie entre 0 et 1F800h.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Registre 7 :	TC3	TC2	TC1	TC0	BD3	BD2	BD1	BD0

BD0 à BD3 = Donnent la couleur du fond dans tous les mode d'écran.

TC0 à TC3 = Donnent la couleur du texte dans les modes textes.

En SCREEN 8 à 12, tous les bits définissent la couleur de font.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 8 :	MS	LP	TP	CB	VR	0	SPD	BW	(V9938 à V9958)

BW = 1 pour régler l'affichage en noir et blanc. Sinon l'affichage se fait en couleur.

SPD = 1 pour désactiver l'affichage des Sprites. Les commandes du VDP travaillent un peu plus vite lorsque les Sprites sont désactivés.

VR = Défini le type de mémoire vidéo (« Video Ram ») :

1 = 64Ko × 1 bit ou bien 64Ko × 4 bits ;

0 = 16 Ko × 1 bit ou bien 16 Ko × 4 bits.

CB = Direction du bus de couleur (« Color Bus ») du VDP. 1 pour entrée ; 0 pour sortie.

TP = 1 pour couleur 0 redéfinissable, par exemple, avec l'instruction COLOR=(0, R, G, B) .

0 pour couleur 0 « transparente ». Ceci permet d'afficher la vidéo en entrée à la place de la couleur n°0 sur les MSX dotés d'une entrée vidéo.

En Basic, on modifie l'état de TP par :

VDP (9) =VDP (9) OR &H20 pour mettre TP à 1

VDP (9) =VDP (9) AND &HDF pour mettre TP à 0

LP = 1 pour activer de mode crayon optique. (Inutilisé sur MSX, mettre à 0.) (N'existe pas sur le V9958.)

MS = 1 pour activer de mode souris. (Inutilisé sur MSX, mettre à 0.) (N'existe pas sur le V9958.)

bit 7 bit 6 bit 5 bit 4 bit 3 bit 2 bit 1 bit 0
 Registre 9 :

LN	0	S1	S0	IL	E0	\overline{NT}	DC
----	---	----	----	----	----	-----------------	----

 (V9938 à V9958)

DC = 1 met la broche \overline{DTCLK} (horloge de base résolution) en mode entrée. 0 met en mode sortie. (Utilisé sur les MSX avec entrée vidéo.)

\overline{NT} = 1 met l'affichage en PAL (313 lignes, 50 Hertz) autrement, affichage en NTSC (256 lignes, 60 Hertz). (Sortie RGB)

E0 = Dans un mode graphique de SCREEN 5 à 12, ce bit sert à activer l'affichage de 2 pages en alternance, 1 frame sur deux, pour la deuxième période. (Pour plus de précisions, voir le registre 13.)

IL = 1 pour affichage entrelacé de 2 pages. 0 pour affichage non-entrelacé.

S0 et S1 = Affichage. 0 = Normal ; 1 = Numérisation ; incrustation, etc ; 2 = Vidéo externe.

LN = 1 pour régler la hauteur de l'écran à 212 points, sinon 192 points.

bit 7 bit 6 bit 5 bit 4 bit 3 bit 2 bit 1 bit 0
 Registre 10 :

0	0	0	0	0	C16	C15	C14
---	---	---	---	---	-----	-----	-----

 (V9938 à V9958)

C14 à C16 = Détermine les 3 bits de poids fort de l'adresse du début de la table des couleurs (qui en comporte 17). Voir le registre 3 pour plus de précisions.

bit 7 bit 6 bit 5 bit 4 bit 3 bit 2 bit 1 bit 0
 Registre 11 :

0	0	0	0	0	0	S16	S15
---	---	---	---	---	---	-----	-----

 (V9938 à V9958)

S15 et S16 = Détermine les 2 bits de poids fort de l'adresse du début de la table des attributs de Sprites (qui en comporte 17). Voir le registre 5 pour plus de précisions.

bit 7 bit 6 bit 5 bit 4 bit 3 bit 2 bit 1 bit 0
 Registre 12 :

T23	T22	T21	T20	BC3	BC2	BC1	BC0
-----	-----	-----	-----	-----	-----	-----	-----

 (V9938 à V9958)

Couleur du texte utilisée pendant la seconde période lors d'un affichage périodique en mode texte 80 colonnes (SCREEN 0). Voir le registre 13 pour plus de précisions.

BC0 à BC3 = Couleur de fond du texte de la deuxième période.

T20 à T23 = Couleur du texte de la deuxième période.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
ON3	ON2	ON1	ON0	OF3	OF2	OF1	OF0	(V9938 à V9958)

OF0 à OF3 = Règle le temps de la seconde période.

ON0 à ON3 = Règle le temps de la première période et active l'affichage périodique lorsque un de ces bits est à 1.

Longueur d'une période en secondes pour une fréquence de 50 Hz :

0000 = 0,0	0100 = 0,8	1000 = 1,6	1100 = 2,4
0001 = 0,2	0101 = 1,0	1001 = 1,8	1101 = 2,6
0010 = 0,4	0110 = 1,2	1010 = 2,0	1110 = 2,8
0011 = 0,6	0111 = 1,4	1011 = 2,2	1111 = 3,0

En mode texte 80 colonnes, ce registre sert à paramétrer les temps des périodes du clignotement des couleurs de texte. La couleur de texte définie par le registre 13 s'affiche pendant la première période. La couleur actuelle s'affiche pendant la deuxième période.

Marche à suivre en mode texte (80 colonnes seulement) :

1. Charger le registre 7 avec les couleurs du texte pour la seconde période.
2. Charger le registre 12 avec les couleurs du texte pour la première période.
3. Modifier la table des couleurs en VRAM. Cette table occupe 240 octets (0800h à 08EFh par défaut), chaque bit correspond à caractère (24 lignes × 80 colonnes = 1920 caractères d'où 1920 bits, soit 240 octets). Si le bit est à 1, le caractère clignote avec les couleurs définies par les registres 7 et 12 alors que s'il se trouve à 0, le caractère ne clignote pas. Il garde la couleur du registre 7.
4. Régler le registre 13 avec les temps d'affichage adéquats.

Voici un exemple en Basic :

```
10 SCREEN 0:WIDTH80
20 COLOR 10,0,0:FOR I=&H800+239:VPOKE I,0:NEXT
30 VDP(13)=&H10:VDP(14)=&H44 ' registre 12 = 010H et reg. 13 = 044H
40 PRINT"Ce petit programme permet de faire clignoter automatiquement un
texte !":PRINT:PRINT:VPOKE &H804,&H1F:VPOKE &H805,&HF0
```

Après avoir lancé le programme ci-dessus, vous pouvez continuer à travailler. Essayez de changer le &H44 de la ligne 30 en &H11 et refaites RUN. Vous avez changé la vitesse dans le registre 13. Mettez maintenant &HBB à la place de &H11, en même temps, changez le &H10 en &H67. Le mot apparaît à présent en jaune sur noir (comme le reste) puis en rouge sur fond cyan. Vous avez agi sur le registre 12. Enfin, transformez le VPOKE &H805,&HF0 en VPOKE &H805,&HFF. A l'exécution, vous observerez que vous avez modifié des indicateurs et que de nouveaux caractères clignotent.

En mode graphique, ce registre sert à paramétrer un cycle de deux périodes d'affichage d'une page paire ou impaire. La page paire inférieure s'affiche pendant première période du cycle. La page impaire ou, l'alternance des deux pages si le bit E0 du registre 9 à 1, s'affiche pendant la seconde période.

Marche à suivre en mode graphique (SCREEN 5 à 12) :

Les pages graphiques vont par deux. En SCREEN 5 et 6, on peut choisir de faire afficher en alternance les pages 0 et 1 ou alors 2 et 3 mais jamais 0 et 3, 1 et 3 ou 0 et 2.

1. Paramétrer les bits N10 à N14 du registre 2, afin d'afficher une page paire.
2. Régler les temps d'affichages respectifs et activer l'affichage périodique grâce au registre 13.
3. Mettre le bit E0 du registre 9 à 1 pour activer l'alternance de 2 page sur la seconde période.

Exemple d'utilisation en Basic :

```
10 COLOR 10,0,0:SCREEN 5
20 ON STOP GOSUB 190:STOP ON
30 PI=3.141592654#
40 CIRCLE(100,100),50,7,PI/2,2*PI
50 LINE(50,100)-(150,100),7:LINE(100,50)-(100,150),7:PAINT
(98,98),10,7:PAINT(102,102),6,7:PAINT(98,102),12,7
60 SET PAGE 1,1:CLS
70 CIRCLE(100,100),50,7,PI/4,2*PI-PI/4:LINE(65,65)-(135,135),7:LINE(65,135)-
(135,65),7:PAINT(100,98),10,7:PAINT(100,102),6,7:PAINT(98,100),12,7
80 'VDP(2)=&H3F
90 A$=INKEY$
100 IF A$="1" THEN VDP(14)=&H33:VDP(10)=VDP(10) OR 4
110 IF A$="2" THEN VDP(14)=&H3:VDP(10)=VDP(10) OR 4
120 IF A$="3" THEN VDP(14)=&H30:VDP(10)=VDP(10) OR 4
130 IF A$="4" THEN VDP(14)=&H0:VDP(10)=VDP(10) OR 4
140 IF A$="5" THEN VDP(14)=&H33:VDP(10)=VDP(10) AND 251
150 IF A$="6" THEN VDP(14)=&H3:VDP(10)=VDP(10) AND 251
160 IF A$="7" THEN VDP(14)=&H30:VDP(10)=VDP(10) AND 251
170 IF A$="8" THEN VDP(14)=&H0:VDP(10)=VDP(10) AND 251
180 GOTO 90
190 STOP OFF:VDP(14)=0:VDP(10)=VDP(10) AND 251:END
```

Les lignes 30 à 70 se chargent de faire un joli dessin sur la page 0 et la 1. Les lignes 20 et 190 désactive l'affichage périodique et alterné automatique.

Lorsqu'on presse une touche de 1 à 8, voici ce qui se passe :

- 1 = La page 0 s'affiche pendant la première période puis les pages 1 et 0 s'affichent une frame sur deux pendant la deuxième période.
- 2 = Les pages 1 et 0 s'affichent une frame sur deux continuellement. L'affichage périodique est désactivé.
- 3 = La page 0 s'affiche pendant la première période. Il n'y a pas de deuxième période donc la page 0 reste toujours affichée.
- 4 = Les pages 1 et 0 s'affichent une frame sur deux continuellement. L'affichage périodiques est désactivé.
- 5 = Affiche la page 0 pendant la première période puis affiche la page 1 pendant la deuxième période.
- 6 = Affiche la page 1. L'affichage périodique est désactivé.
- 7 = Affiche la page 0 pendant la première période. Il n'y a pas de deuxième période donc la page 0 reste toujours affichée.
- 8 = Affiche la page 1. L'affichage périodique est désactivé. (paramètres par défaut)

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 14 :	0	0	0	0	0	V16	V15	V14	(V9938 à V9958)

V14 à V16 = Ce registre permet d'accéder aux adresses supérieures à 03FFFh (16Ko) de la VRAM. Ces bits sont les 3 bits de poids fort de l'adresse de la VRAM à laquelle on désire accéder. Voir le paragraphe 6.5 « [Lecture et écriture dans la mémoire vidéo](#) ».

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 15 :	0	0	0	0	ST3	ST2	ST1	ST0	(V9938 à V9958)

ST0 à ST3 = Indiquent le numéro de registre de statut que l'on désire lire (0 ~ 9). Voir le paragraphe 6.4 « Comment accéder aux registres du VDP » rubrique « [Lire un registre de statut \(STATUS REGISTER\)](#) » pour plus de précisions.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 16 :	0	0	0	0	CC3	CC2	CC1	CC0	(V9938 à V9958)

CC0 à CC3 = Déterminent le numéro de la couleur lors d'un accès palette (0 ~ 15). Voir le paragraphe 6.4 « Comment accéder aux registres du VDP » rubrique « [Ecrire dans un registre de la palette des couleurs](#) » pour plus de précisions.

Valeur de la palette de chaque couleur par défaut :

Numéro de couleur	R	V	B
0	0	0	0
1	0	0	0
2	1	6	1
3	3	7	3
4	1	1	7
5	2	3	7
6	5	1	1
7	2	6	7

Numéro de couleur	R	V	B
8	7	1	1
9	7	3	3
10	6	6	1
11	6	6	4
12	1	4	1
13	6	2	5
14	5	5	5
15	7	7	7

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 17 :	AII	0	RS5	RS4	RS3	RS2	RS1	RS0	(V9938 à V9958)

Ce registre est utilisé lors d'un accès indirect à un autre registre du VDP.

RS0 à RS5 = Numéro de registre désiré du VDP.

AII = Mode auto-incrémentation OFF.

La donnée du registre s'écrira sur le port 3 du VDP. En mode auto-incrémentation (AII à 0), chaque donnée que l'on écrit sur le port 3 provoque une incrémentation du numéro de registre contenu dans le registre 17. Cela permet, par exemple, d'envoyer des données dans une série de registre.

Voir le paragraphe 6.4 « Comment accéder au VDP » rubrique « [Ecrire dans un registre de contrôle](#) », pour plus de précisions.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 18 :	V3	V2	V1	V0	H3	H2	H1	H0	(V9938 à V9958)

H0 à H3 = Ajustement horizontal de l'affichage de l'écran.

7 - 6 - 5 -	...	- 1 - 0 - 15 - 14 -	...	- 10 - 9 - 8
gauche		centre		droite

V0 à V3 = Ajustement vertical de l'affichage de l'écran.

8 - 7 - 6 -	...	- 1 - 0 - 15 - 14 -	...	- 9 - 8 - 7
bas		centre		haut

Ce registre est équivalent au SET ADJUST du Basic mais sans sauvegarde du paramètre dans la SRAM.

Note : Veuillez attendre que le bit CE du registre 2 de lecture soit à 0 avant d'accéder à ce registre lorsque l'affichage de l'écran ou des Sprites est désactivé.

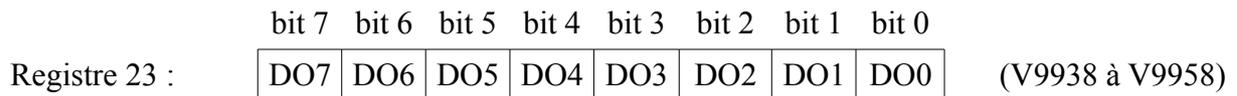
	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 19 :	IL7	IL6	IL5	IL4	IL3	IL2	IL1	IL0	(V9938 à V9958)

IL0 à IL7 = Défini le numéro de ligne où une interruption programmée doit se produire lorsque le bit 4 (IE1) du registre 0 est à 1.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 20 :	0	0	0	0	0	0	0	0	(V9938)
Registre 21 :	0	0	1	1	1	0	1	1	(V9938)
Registre 22 :	0	0	0	0	0	1	0	1	(V9938)

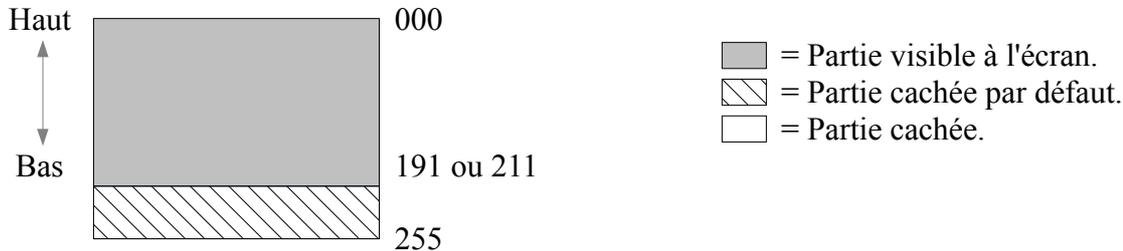
Les valeurs de ces 3 registres sont indiquent la fréquence « Color burst » du signal composite. Ne pas les modifier. Seule exception à la règle, on peut mettre ces 3 registres à 0 pour couper la sortie composite. Il suffit de remettre les 3 registres à leur valeur initiale pour rétablir la sortie composite.

Note : Beaucoup de MSX2 n'utilisent pas cette sortie car ils utilisent un dispositif externe pour faire la sortie composite à partir du signal RGB. De plus, le V9958 n'a pas de sortie composite.

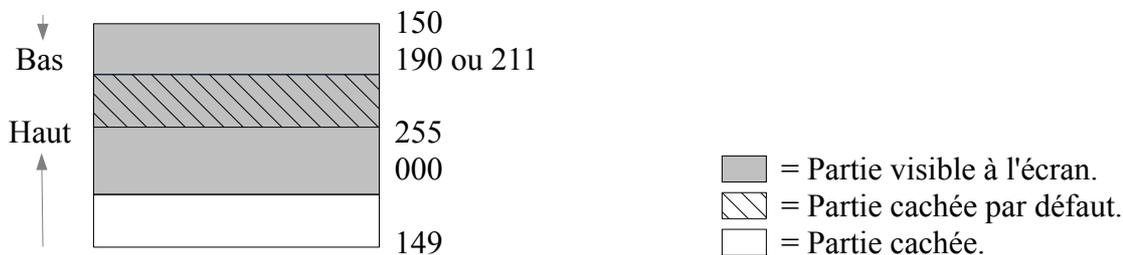


D0 à D7 = Défini le décalage vertical de l'affichage.

Affichage d'une page sans décalage :



Exemple avec 150 dans le registre 23 :



Note : Ce registre permet de réaliser très facilement des scrollings verticaux. Voici un petit programme Basic pour illustrer cette facilité :

```

10 SCREEN 8
20 CIRCLE(100,100),50,200
30 FOR I=0 TO 50
40 VDP(24)=I ' (équivalent à C=PEEK(7)+1: OUT C,I: OUT C,&H80+23)
50 NEXT
60 FOR I=50 TO 0 STEP -1
70 VDP(24)=I
80 NEXT
90 GOTO 30

```

L'équivalent en assembleur serait :

```

                ORG 0C000H
;
EXTROM         EQU 0015FH
CHGMOD        EQU 000D1H
CHGCLR        EQU 00062H
CLS           EQU 000C3H
NVBXLN        EQU 000C9H
BREAKX        EQU 000B7H
TOTEXT        EQU 000D2H
;
FORCLR        EQU 0F3E9H
BAKCLR        EQU 0F3EAH
BDRCLR        EQU 0F3EBH
GXPOS         EQU 0FCB3H
GYPOS         EQU 0FCB5H
ATRBYT        EQU 0F3F2H

```

```

LOGOPR      EQU 0FB02H
;
DEBUT:      LD A,8
            LD IX,CHGMOD
            CALL EXTROM      ; Initialisation (Ecran, Couleurs, Etc)
;
            LD A,00AH
            LD (FORCLR),A
            LD A,0
            LD (BAKCLR),A
            LD (BDRCLR),A
            CALL CLS        ; efface l'écran
;
            LD BC,060H
            LD DE,060H
            LD HL,090H
            LD (GXPOS),HL
            LD (GYPOS),HL
            LD A,0FCH
            LD (ATRBYT),A
            LD (LOGOPR),A
            LD IX,NVBXLN
            CALL EXTROM      ; Trace un rectangle
;
            LD A,(7)         ; C = Numéro du port E/S correspondant
            LD C,A          ; au port 0 d'écriture du VDP.
            INC C           ; Port E/S du port 1 d'écriture
;
; Boucle Principale
;
LOOP:        CALL UP
            CALL BREAKX      ; Test du CTRL+STOP
            JR C,OUT
            CALL DOWN
            CALL BREAKX      ; Test du CTRL+STOP
            JR C,OUT         ; Sortie du programme si CTRL+STOP
            JP LOOP
;
OUT:         CALL TOTEXT     ; Retour au mode texte
            RET
;
UP:         LD B,032H
LOOP1:      LD A,032H
            SUB B
            CALL VDP
            CALL WAIT
            DJNZ LOOP1      ; saut à LOOP1: tant que B n'est pas à 0
            RET
;
DOWN:       LD B,032H
LOOP2:      LD A,B
            CALL VDP
            CALL WAIT
            DJNZ LOOP2      ; saut à LOOP2: tant que B n'est pas à 0
            RET
;
VDP:        DI              ; Désactive les interruptions
            OUT (C),A
            LD A,080H+23

```

```
      OUT (C),A
      EI           ; Active les interruptions
      RET

;
WAIT:  LD HL,002FFH
HEP:   DEC HL
      LD A,H
      OR L
      JR NZ,HEP   ; saut à HEP: tant que HL n'est pas à 0
      RET
```

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 25 :	0	CMD	VDS	YAE	YJK	WTE	MSK	SP2	(V9958)

SP2 = 0 pour scrolling horizontal de l'écran sur une page (valeur par défaut),

1 pour scrolling horizontal de l'écran sur deux page.

MSK = Active un masque de 8 ou 16 lignes sur la partie gauche de l'écran.

WTE = 0 désactive la fonction « Wait » (le VDP fonctionne comme un V9938),

1 active la fonction « Wait ». Met en attente le CPU lorsqu'un accès du CPU à la VRAM est effectué. (Inutilisé sur MSX.)

YJK = 1 pour configurer la table Bitmap du SCREEN 8 au format YJK au lieu de RGB. Cela permet d'afficher jusqu'à 19268 couleurs à l'écran au lieu de 256. Les signaux de sortie sont tout de même convertis en RGB (sur 5 bits) et affiché en analogique. La palette des couleurs d'affichage du Sprite reste au format RGB.

Format de la table des formes en mode YJK pour chaque ligne de 4 pixels :

	bit 7	bit 6	bit 5	bit 4	bit 2	bit 2	bit 1	bit 0
Premier octet :	Y ₁				bits de poids faible de K			
Second octet :	Y ₂				bits de poids fort de K			
Troisième octet :	Y ₃				bits de poids faible de J			
Quatrième octet :	Y ₄				bits de poids fort de J			

La couleur du premier octet est codée par Y₁, J et K. Le second par Y₂, J et K, le troisième par Y₃, J et K puis le quatrième par Y₄, J et K. Et ainsi de suite...

Formules de conversion :

$$\begin{aligned}
 R &= Y + J & Y &= (2R + G + 4B) / 8 \\
 G &= Y + K & J &= (6R - G + 4B) / 8 \\
 B &= 1,25Y - 0,5J - 0,25K & K &= (-2R + 7G - 4B) / 8
 \end{aligned}$$

YAE = 1 configure la table Bitmap en mode YAE. (Mettre aussi YJK à 1.) Cela donne le mode SCREEN 10/11. C'est à dire qu'un bit attribue à chaque pixel le mode dans lequel il doit s'afficher, RVB ou YJK (aux mêmes caractéristiques que le SCREEN 5).

Format de la table des formes en mode YAE pour chaque ligne de 4 pixels :

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Premier octet :	Y ₁			A ₁	bits de poids faible de K			
Second octet :	Y ₂			A ₂	bits de poids fort de K			
Troisième octet :	Y ₃			A ₃	bits de poids faible de J			
Quatrième octet :	Y ₄			A ₄	bits de poids fort de J			

La couleur du premier octet est codée par Y₁, J et K. Le second par Y₂, J et K, le troisième par Y₃, J et K puis le quatrième par Y₄, J et K. Et ainsi de suite... Lorsque l'attribut d'un octet est à 1, J et K sont ignorés. Y₁ ~ Y₄ devient le numéro de couleur comme en SCREEN 5.

VDS = Ce bit détermine la fonction de la broche 8 du VDP. Le bit VDS est à zéro, il s'agit d'un signal d'horloge à 3,579545 MHz pour le Z80A. Sinon, il s'agit du signal VDS en sortie. (Laisser ce bit à 0)

CMD = Ce bit permet d'utiliser les commandes du VDP dans tous les modes d'écran. (0 par défaut.)

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 26 :	0	0	HO8	HO7	HO6	HO5	HO4	HO3	(V9958)
Registre 27 :	0	0	0	0	0	HO2	HO1	HO0	(V9958)

HO0 à HO8 = Numéro de ligne verticale qui se trouvera le plus à gauche (le bit HO8 n'est utile que si SP2 du registre 25 est à 1. En screen 6 et 7, le défilement se fera de 2 en 2 lignes)

Les quinze derniers registres du VDP sont utilisés pour l'exécution des commandes graphiques. Voir le paragraphe 6.8 « [Les commandes internes du VDP](#) » pour plus de précisions.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 32 :	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	(V9938 à V9958)
Registre 33 :	0	0	0	0	0	0	0	SX8	(V9938 à V9958)

SX0 à SX8 = Codent l'abscisse du point d'origine ($0 < X < 511$)

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 34 :	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	(V9938 à V9958)
Registre 35 :	0	0	0	0	0	0	SY9	SY8	(V9938 à V9958)

SY0 à SY9 = Codent l'ordonnée du point d'origine ($0 < Y < 1023$)

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 36 :	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	(V9938 à V9958)
Registre 37 :	0	0	0	0	0	0	0	DX8	(V9938 à V9958)

DX0 à DX8 = Abscisse du premier pixel de destination ($0 < X < 511$)

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 38 :	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	(V9938 à V9958)
Registre 39 :	0	0	0	0	0	0	DY9	DY8	(V9938 à V9958)

DY0 à DY9 = Codent l'ordonnée du point de destination ($0 < Y < 1023$)

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 40 :	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	(V9938 à V9958)
Registre 41 :	0	0	0	0	0	0	0	NX8	(V9938 à V9958)

NX0 à NX8 = Déplacement horizontal ($0 < NX < 511$).

Note : Ce registre se comporte différemment avec la commande LINE.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 42 :	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	(V9938 à V9958)
Registre 43 :	0	0	0	0	0	0	NY9	NY8	(V9938 à V9958)

NY0 à NY9 = Déplacement vertical ($0 < NY < 1023$).

Note : Ce registre se comporte différemment avec la commande LINE.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 44 :	CL7	CL6	CL5	CL4	CL3	CL2	CL1	CL0	(V9938 à V9958)

CL0 à CL7 = Codent la couleur.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 45 :	0	MXC	MXD	MXS	DIY	DIX	EQ	MAJ	(V9938 à V9958)

Registre de données pour les commandes du VDP. Nous verrons ces bits en détail dans le chapitre 6.8 « Les commandes du VDP ».

MAJ = Déplacement le plus long. 0 pour horizontal ; 1 pour vertical. (Utilisé uniquement par la commande LINE.)

EQ = 0 pour stopper la recherche lorsque la couleur est trouvée ; 1 pour stopper la recherche lorsque la couleur est différente de celle paramétrée.

DIX = Lorsque ce bit est à 1, la valeur de déplacement horizontal (registre 40 et 41) est considérée négative. Le déplacement se fait dans l'autre sens.

DIY = Lorsque ce bit est à 1, la valeur de déplacement vertical (registre 42 et 43) est considérée négative. Le déplacement se fait dans l'autre sens.

MXS = Mémoire vidéo source. 0 pour VRAM principale ; 1 pour VRAM étendue (pour les MSX avec 192 Ko de VRAM).

MXD = Mémoire vidéo de destination. 0 pour VRAM principale ; 1 pour VRAM étendue (pour les MSX avec 192 Ko de VRAM).

MXC = Commutation RAM/VRAM. 1 pour extension de RAM ; 0 pour VRAM. Inutilisé sur MSX, mettre à 0.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 46 :	CM3	CM2	CM1	CM0	LO3	LO2	LO1	LO0	(V9938 à V9958)

LO0 à LO3 = Définissent le code de l'opérateur logique à utiliser.

CM0 à CM3 = Définissent la commande du VDP à exécuter.

6.7 Les registres de statut du VDP

Voici la liste complète de tous les registres de statut, qui ne sont accessibles qu'en lecture pour l'utilisateur, avec la signification des informations qu'ils contiennent.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 0 :	F	5S	C	SN4	SN3	SN2	SN1	SN0	

SN0 à SN4 = Donnent le numéro du 5ème Sprite (ou du 9ème dans les SCREEN 5 à 8) lorsque le bit 5S se trouve à 1.

C = Ce bit passe à 1 lorsque deux Sprites entrent en collision.

5S = Indicateur qui passe à 1 lorsque 5 Sprites (9 dans les SCREEN 4 à 12) se trouvent sur une même ligne.

F = Indicateur de l'interruption qui se produit à chaque fin d'affichage de l'écran. (Vertical blank interrupt.) Chaque fois que le registre est lu, ce bit repasse à 0.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 1 :	FL	LPS	ID4	ID3	ID2	ID1	ID0	FH	(V9938 à V9958)

FH = Indicateur de l'interruption qui se produit avant le balayage de la ligne horizontale indiquée au registre 19.

ID0 à ID3 = Donnent le numéro d'identification du processeur vidéo. 00000 pour V9938 ; 00010 pour V9958.

LPS = Indicateur du bouton du crayon optique ou du bouton gauche de la souris. Ce bit passe à 1 lorsque le bouton est pressé. (Inutilisé sur MSX.) (N'existe pas sur le V9958.)

FL = Indicateur du crayon optique ou du bouton gauche de la souris. Pour que ce dernier fonctionne, il faut que le bit IE2 soit à 1. Ce bit passe à 1 lorsque le crayon optique détecte de la lumière ou bien lorsque le bouton est pressé. Ce bit est automatiquement remis à 0 quand on lit le registre 1. (Inutilisé sur MSX.) (N'existe pas sur le V9958.)

Note : pour plus de renseignements sur le fonctionnement du crayon optique et de la souris, voir le paragraphe 6.11 « [La souris et le crayon optique](#) »

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 2 :	TR	VR	HR	BD	1	1	EO	CE	(V9938 à V9958)

CE = Ce bit est à 1, lorsque le VDP est en train d'exécuter une instruction.

EO = Indicateur de période. 0 pour première période; 1 = Pour deuxième période.

BD = Lors d'une commande de recherche, ce bit passe à 1 lorsque la couleur de frontière est trouvée.

HR = Lors d'un balayage d'une horizontale, ce bit est à 1.

VR = Lors du balayage de l'écran, ce bit est à 1.

TR = Lorsque ce bit est à 1, le VDP peut effectuer des transferts du CPU à la VRAM.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 3 :	X7	X6	X5	X4	X3	X2	X1	X0	(V9938 à V9958)
Registre 4 :	1	1	1	1	1	1	1	X8	(V9938 à V9958)

X0 à X8 = Indiquent l'abscisse du point de collision des Sprites.

(Sur le V9938, ces bits peuvent indiquer aussi l'abscisse du point où pointe le crayon optique ou bien, le déplacement horizontal relatif de la souris mais ces modes ne sont pas utilisés sur MSX.)

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 5 :	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	(V9938 à V9958)
Registre 6 :	1	1	1	1	1	1	Y9	Y8	(V9938 à V9958)

Y0 à Y9 = Indiquent l'ordonnée du point de collision des Sprites.

(Sur le V9938, ces bits peuvent indiquer aussi l'ordonnée du point où pointe le crayon optique ou bien, le déplacement vertical relatif de la souris mais ces modes ne sont pas utilisés sur MSX.)

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 7 :	C7	C6	C5	C4	C3	C2	C1	C0	(V9938 à V9958)

C0 à C7 = Utilisés par les commandes POINT et LMCM.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 8 :	BX7	BX6	BX5	BX4	BX3	BX2	BX1	BX0	(V9938 à V9958)
Registre 9 :	1	1	1	1	1	1	1	BX8	(V9938 à V9958)

BX0 à BX8 = Indiquent l'abscisse du premier point trouvé lors d'une recherche avec la commande SRCH du VDP.

6.8 Les commandes internes du V9938 et V9958.

Ces VDP disposent d'un jeu de commandes qui permettent de copier une zone de mémoire vers une autre, de tracer quelques formes simples, etc. Dans cette partie, nous allons détailler ces commandes une à une, et voir comment on les met en œuvre.

Il y a en tout 12 commandes distinctes plus une commande d'arrêt (« STOP »).

Voici un tableau récapitulatif :

Mnémonique	Source	Destination	Signification
HMMC	Z80	VRAM	High speed Move to Memory from CPU
YMMM	VRAM	VRAM	Y Move to Memory from Memory
HMMM	VRAM	VRAM	High speed Move to memory from memory
HMMV	VDP	VRAM	High speed Move to Memory from VDP
LMMC	Z80	VRAM	Logical Move to Memory from CPU
LMCM	VRAM	Z80	Logical Move to CPU from Memory
LMMM	VRAM	VRAM	Logical Move to Memory from memory
LMMV	VDP	VRAM	Logical Move to Memory from VDV
LINE	VDP	VRAM	LINE
SRCH	VDP	VRAM	SeaRCH
PSET	VDP	VRAM	Point SET
POINT	VRAM	VDP	is POINT set ?
STOP	-	-	STOP

Note : Il est nécessaire de vérifier que le bit CE du registre de statut 2 soit à 0 avant d'exécuter une de ces commandes.

A partir du SCREEN 5, le VDP travaille uniquement en coordonnées X et Y. Les notions d'adresse mémoire et de page n'existent pas. Tout se passe comme si le VDP opérait sur un écran géant de 256x1024 (ou 512x1024). La partie visible ne serait alors qu'une fenêtre suivant le schéma :

SCREEN 5		VRAM	SCREEN 6	
0, 0	255, 0	00000H	0, 0	511, 0
page 0			page 0	
0, 255	255, 255	07FFFH	0, 255	511, 255
0, 256	255,256	08000H	0, 256	511, 256
page 1			page 1	
0, 511	255,511	0FFFFH	0, 511	511, 511
0, 512	255,512	10000H	0, 512	511, 512
page 2			page 2	
0, 767	255, 767	17FFFH	0, 767	511, 767
0, 768	255, 768	18000H	0, 768	511, 768
page 3			page 3	
0, 1023	255, 1023	1FFFFH	0, 1023	511, 1023

SCREEN 7		VRAM	SCREEN 8 à 12	
0, 0	511, 0	00000H	0, 0	255, 0
page 0			page 0	
0, 255	511, 255	0FFFFH	0, 255	255, 255
0, 256	511, 256	10000H	0, 256	255,256
page 1			page 1	
0, 511	511, 511	1FFFFH	0, 511	255,511

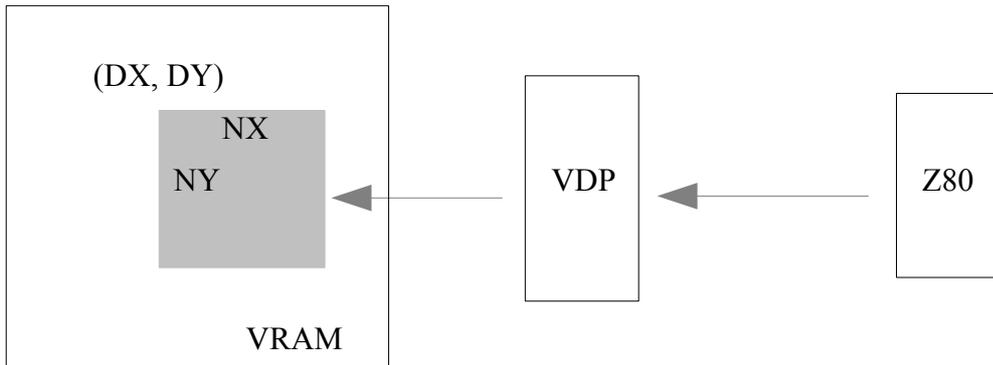
Il est possible d'exécuter certaines commandes en utilisant un opérateur logique qui s'effectue avec les pixels affichées sur la zone à tracer grâce aux bits LO0 à LO3 du registre 45.

Opérateur	Code	Fonction
IMP	0000	La couleur affichée sera remplacée par la nouvelle.
AND	0001	Couleur résultante = (couleur affichée) ET (nouvelle couleur).
OR	0010	Couleur résultante = (couleur affichée) OU (nouvelle couleur).
XOR	0011	Couleur résultante = (couleur affichée) XOR (nouvelle couleur).
NOT	0100	La couleur affichée sera remplacée par la nouvelle.

Opérateur	Code	Fonction
TIMP	1000	Idem à IMP mais si la couleur source est 0, celui-ci restera 0.
TAND	1001	Idem à AND mais si la couleur source est 0, celui-ci restera 0.
TOR	1010	Idem à OR mais si la couleur source est 0, celui-ci restera 0.
TXOR	1011	Idem à XOR mais si la couleur source est 0, celui-ci restera 0.
TNOT	1100	Idem à NOT mais si la couleur source est 0, celui-ci restera 0.

HMMC (High speed Move to Memory from CPU)

Schéma :



Fonction : HMMC permet de remplir une zone rectangulaire à l'écran avec des données provenant du Z80.

Source : Z80.

Destination : VRAM.

Voici la marche à suivre pour exécuter la commande HMMC :

1. Écrire les paramètres dans les registres suivants.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 36 :	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	abscisse destination
Registre 37 :	0	0	0	0	0	0	0	DX8	(0 à 511)
Registre 38 :	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	ordonnée destination
Registre 39 :	0	0	0	0	0	0	DY9	DY8	(0 à 1023)

DX0 à DX8 = Abscisse du premier pixel de destination.

En SCREEN 5 et 7, le bit de poids faible est perdu.

En SCREEN 6, les bits DX0 et DX1 sont perdus.

DY0 à DY9 = Ordonnée du premier pixel de destination.

Registre 40 :	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	déplacement horizontal (0 à 511)
Registre 41 :	0	0	0	0	0	0	0	NX8	
Registre 42 :	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	déplacement vertical (0 à 1023)
Registre 43 :	0	0	0	0	0	0	NY9	NY8	

NX0 à NX8 = Nombre de pixels à placer dans la direction horizontale (en largeur).

En SCREEN 5 et 7, NX0 est perdu.

En SCREEN 6, NX0 et NX1 sont perdus.

NY0 à NY9 = Nombre de pixels à placer dans la direction verticale. (hauteur)

Registre 44 :

CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
-----	-----	-----	-----	-----	-----	-----	-----

donnée

SCREEN 5 et 7 :

CR0 à CR3 = Couleur du pixel d'abscisse impair.

CR4 à CR7 = Couleur du pixel d'abscisse pair.

SCREEN 6 :

CR0 et CR1 = Couleur du troisième pixel à droite de celui à l'abscisse multiple de 4.

CR2 et CR3 = Couleur du deuxième pixel à droite de celui à l'abscisse multiple de 4.

CR4 et CR5 = Couleur du pixel à droite de celui dont l'abscisse est multiple de 4.

CR6 et CR7 = Couleur du pixel dont l'abscisse est multiple de 4.

SCREEN 8 :

CR0 à CR7 = Couleur du pixel.

Registre 45 :

0	0	MXD	0	DIY	DIX	0	0
---	---	-----	---	-----	-----	---	---

paramètres

DIX = Direction horizontale pour NX. 0 pour droite ; 1 pour gauche.

DIY = Direction verticale pour NY. 0 pour bas ; 1 pour haut.

MXD = 0 pour sélectionner la VRAM principale.

1 pour accéder la mémoire vidéo étendue sur les MSX avec 192 Ko de VRAM.

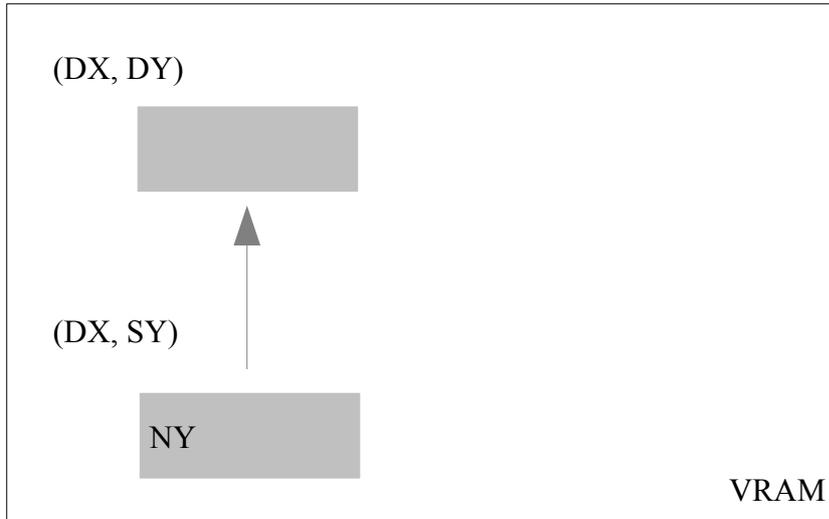
2. Mettre le registre 46 à 11110000B (0F0h) pour exécuter la commande.
3. Lire le registre de statut 2, si le bit CE est à 0, alors le processus est terminé. Sinon, tester l'état du bit TR, si celui-ci est à 0, alors le processeur vidéo n'est pas prêt à recevoir l'octet suivant, recommencer en 3. Si, par contre, ce bit est à 1, continuer en 4.
4. Envoyer l'octet suivant à mettre en VRAM dans le registre 45 et reprendre toute l'opération à l'étape 3.

Une fois la commande terminée, les registres du processeur vidéo se trouveront dans l'état suivant :

R32~33	R34~35	R36~37	R38~39	R40~41	R42~43	R44	R45
idem	idem	idem	modifié	idem	modifié	idem	idem

YMMM (Y Move to Memory from Memory)

Schéma :



Fonction : YMMM autorise la copie d'octets d'une zone de mémoire vidéo à une autre, le point d'origine de cette dernière ayant la même abscisse.

Source : VRAM

Destination : VRAM

Voici la marche à suivre pour exécuter la commande YMMM :

1. Écrire les paramètres dans les registres suivants.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 34 :	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	ordonnée source (0 à 1023)
Registre 35 :	0	0	0	0	0	0	SY9	SY8	

SY0 à SY9 = Ordonnée du pixel d'origine.

Registre 36 :	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	abscisse destination (0 à 511)
Registre 37 :	0	0	0	0	0	0	0	DX8	
Registre 38 :	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	ordonnée destination (0 à 1023)
Registre 39 :	0	0	0	0	0	0	DY9	DY8	

DX0 à DX8 = Abscisse du premier pixel de destination.

En SCREEN 5 et 7, le bit de poids faible est perdu.

En SCREEN 6, les bits DX0 et DX1 sont perdus.

DY0 à DY9 = Ordonnée du pixel de destination.

Registre 42 :	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	déplacement vertical
Registre 43 :	0	0	0	0	0	0	NY9	NY8	(0 à 1023)

NY0 à NY9 = Nombre de pixels à copier dans la direction verticale.

Registre 45 :	0	0	MXD	0	DIY	DIX	0	0	paramètres
---------------	---	---	-----	---	-----	-----	---	---	------------

DIX = Direction horizontale pour NX. 0 pour droite ; 1 pour gauche.

DIY = Direction verticale pour NY. 0 pour bas ; 1 pour haut.

MXD = 0 pour sélectionner la VRAM principale.

1 pour accéder à la VRAM étendue des MSX ayant 192 Ko de VRAM.

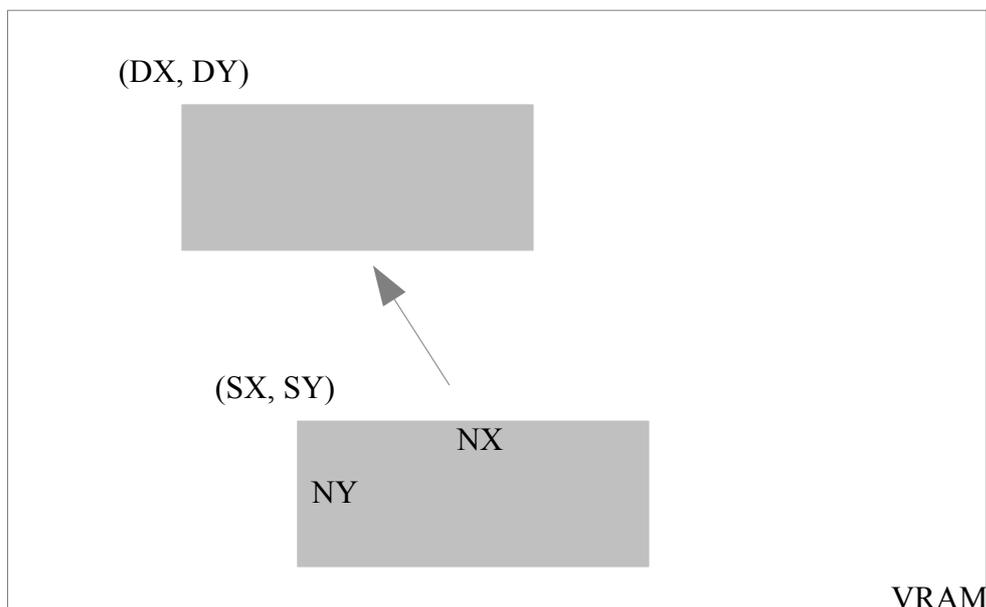
2. Mettre 11100000B (0E0h) dans le registre 46 pour exécuter la commande.
3. Le bit CE du registre de statut 2 restera à 1 tant que le processus n'est pas terminé. Vérifier que ce bit soit à 0 avant d'exécuter la commande suivante ou de changer de paramètre.

Une fois l'instruction terminée, les registres du processeur vidéo se trouveront dans l'état suivant :

32~33	34~35	36~37	38~39	40~41	42~43	44	45
	modifié	idem	modifié		modifié		idem

HMMM (High Speed Move to Memory from Memory)

Schéma :



Fonction : HMMM autorise la copie rapide d'octets d'une zone de mémoire vidéo à une autre de taille équivalente.

Source : VRAM

Destination : VRAM

Voici la marche à suivre pour exécuter la commande HMMM :

1. Écrire les paramètres dans les registres suivants.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 32 :	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	abscisse source (0 à 511)
Registre 33 :	0	0	0	0	0	0	0	SX8	
Registre 34 :	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	ordonnée source (0 à 1023)
Registre 35 :	0	0	0	0	0	0	SY9	SY8	

SX0 à SX8 = Abscisse du pixel d'origine

SY0 à SY9 = Ordonnée du pixel d'origine.

Registre 36 :	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	abscisse destination (0 à 511)
Registre 37 :	0	0	0	0	0	0	0	DX8	
Registre 38 :	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	ordonnée destination (0 à 1023)
Registre 39 :	0	0	0	0	0	0	DY9	DY8	

DX0 à DX8 = Abscisse du premier pixel de destination.

En SCREEN 5 et 7, le bit DY0 est perdu.

En SCREEN 6, les bits DX0 et DX1 sont perdus.

DY0 à DY9 = Ordonnée du pixel de destination.

Registre 40 :	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	déplacement horizontal (0 à 511)
Registre 41 :	0	0	0	0	0	0	0	NX8	
Registre 42 :	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	ordonnée destination (0 à 1023)
Registre 43 :	0	0	0	0	0	0	NY9	NY8	

NX0 à NX8 = Nombre de pixels à copier dans la direction horizontale.

NY0 à NY9 = Nombre de pixels à copier dans la direction verticale.

Registre 45 :	0	0	MXD	MXS	DIY	DIX	0	0	paramètres
---------------	---	---	-----	-----	-----	-----	---	---	------------

DIX = Direction horizontale pour NX. 0 pour droite ; 1 pour gauche.

DIY = Direction verticale pour NY. 0 pour bas ; 1 pour haut.

MXS = Mémoire vidéo source. 0 pour sélectionner la VRAM principale ; 1 pour accéder à la mémoire vidéo étendue sur les MSX2 avec 192 Ko de VRAM.

MXD = Mémoire vidéo de destination. 0 pour sélectionner la VRAM principale ; 1 pour accéder à la mémoire vidéo étendue sur les MSX2 avec 192 Ko de VRAM.

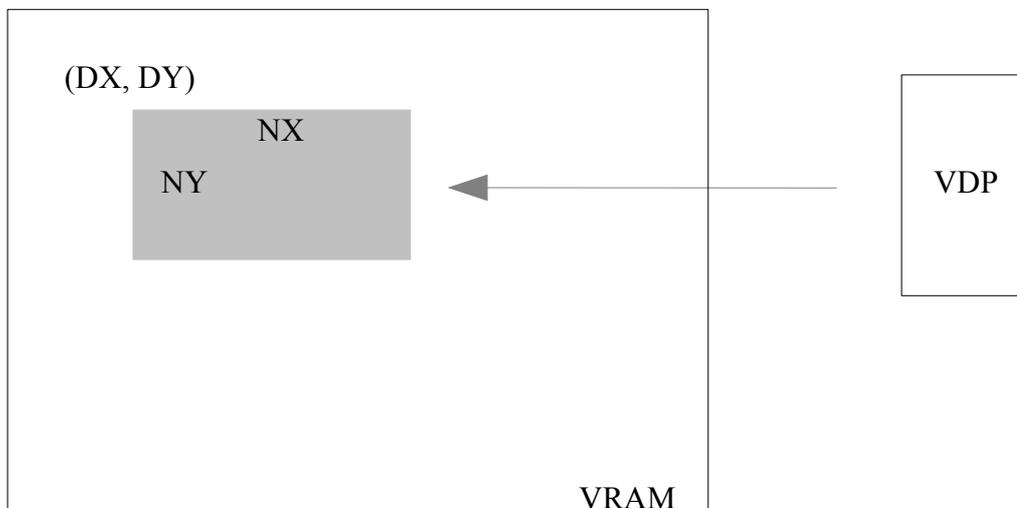
2. Écrire 11010000B (0D0h) dans le registre 46 pour exécuter la commande.
3. Le bit CE du registre de statut 2 restera à 1 tant que le processus n'est pas terminé. Vérifier que ce bit soit à 0 avant d'exécuter la commande suivante ou de changer de paramètre.

Une fois l'instruction terminée, les registres de commande du VDP se trouveront dans l'état suivant :

32~33	34~35	36~37	38~39	40~41	42~43	44	45
idem	modifié	idem	modifié	idem	modifié		idem

HMMV (High speed Move to Memory from VDP)

Schéma :



Fonction : HMMV permet de remplir la mémoire vidéo avec une seule donnée. A la différence des routines du Bios (BIGFIL ou FILVRM), la mémoire vidéo est définie par des coordonnées en X et Y.

Source : VDP.

Destination : VRAM.

Voici la marche à suivre pour exécuter l'instruction HMMV :

1. Écrire les paramètres dans les registres suivants.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Registre 36	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	abscisse
Registre 37	0	0	0	0	0	0	0	DX8	(0 à 511)
Registre 38	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	ordonnée
Registre 39	0	0	0	0	0	0	DY9	DY8	(0 à 1023)

DX0 à DX8 = Abscisse du premier pixel de destination.

En SCREEN 5 et 7, le bit de poids faible est perdu.

En SCREEN 6, les bits DX0 et DX1 sont perdus.

DY0 à DY9 = Ordonnée du pixel origine.

Registre 40 :	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	déplacement horizontal
Registre 41 :	0	0	0	0	0	0	0	NX8	(0 à 511)
Registre 42 :	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	déplacement vertical
Registre 43 :	0	0	0	0	0	0	NY9	NY8	(0 à 1023)

NX0 à NX8 = Nombre de pixels à placer dans la direction horizontale

En SCREEN 5 et 7, le bit NX0 est perdu.

En SCREEN 6, les bits NX0 et NX1 sont perdus.

NY0 à NY9 = Nombre de pixels à placer dans la direction verticale.

Registre 44 :

CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
-----	-----	-----	-----	-----	-----	-----	-----

 donnée

SCREEN 5 et 7 :

CR0 à CR3 = Couleur du pixel d'abscisse impair.

CR4 à CR7 = Couleur du pixel d'abscisse pair.

SCREEN 6 :

CR0 et CR1 = Couleur du troisième pixel à droite de celui à l'abscisse multiple de 4.

CR2 et CR3 = Couleur du deuxième pixel à droite de celui à l'abscisse multiple de 4.

CR4 et CR5 = Couleur du pixel à droite de celui dont l'abscisse est multiple de 4.

CR6 et CR7 = Couleur du pixel dont l'abscisse est multiple de 4.

SCREEN 8 :

CR0 à CR7 = Couleur du pixel.

Registre 45 :

0	0	MXD	0	DIY	DIX	0	0
---	---	-----	---	-----	-----	---	---

 paramètres

DIX = Direction horizontale pour NX. 0 pour droite ; 1 pour gauche.

DIY = Direction verticale pour NY. 0 pour bas ; 1 pour haut.

MXD = Mémoire vidéo de destination. 0 pour sélectionner la VRAM principale ; 1 pour accéder à la mémoire vidéo étendue sur les MSX2 avec 192 Ko de VRAM.

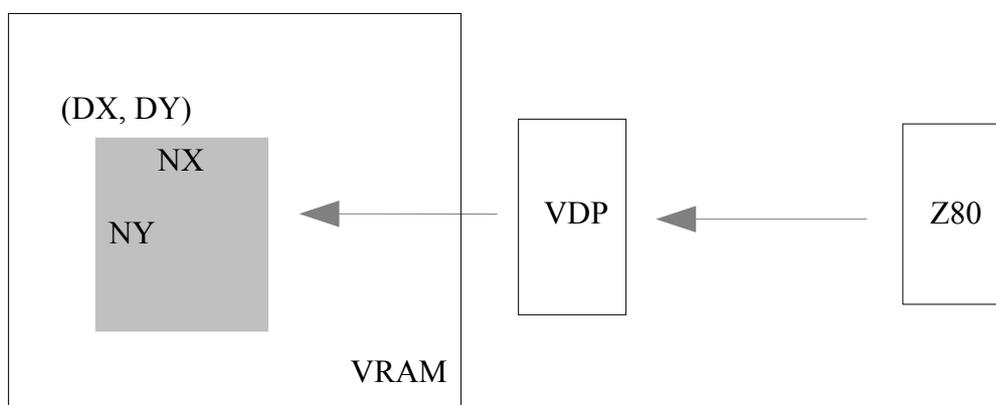
2. Écrire 11000000B (0C0h) dans le registre 46 pour exécuter la commande.
3. Le bit CE du registre de statut 2 restera à 1 tant que le processus n'est pas terminé. Vérifier que ce bit soit à 0 avant d'exécuter la commande suivante ou de changer de paramètre.

Une fois l'instruction terminée, les registres du processeur vidéo se trouveront dans l'état suivant :

32~33	34~35	36~37	38~39	40~41	42~43	44	45
		idem	modifié	idem	modifié	idem	idem

LMMC (Logical Move to Memory from CPU)

Schéma :



Fonction : LMMC permet de remplir la mémoire vidéo avec des données provenant du Z80. Le remplissage se fait pixel par pixel. Cette instruction s'exécute donc moins rapidement que HMMC. En contrepartie, il est possible de définir un opérateur logique. A la différence des routines du Bios (BIGFIL ou FILVRM), les données servant à remplir la mémoire vidéo peuvent être toutes différentes. De plus, la mémoire vidéo est définie par des coordonnées X e Y.

Source : Z80

Destination : VRAM

Voici la marche à suivre pour exécuter l'instruction LMMC :

1. Écrire les paramètres dans les registres suivants.

	bit 7	bit 6	bit 5	bit 4	bit 4	bit 2	bit 1	bit 0	
Registre 32 :	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	abscisse source (0 à 511)
Registre 33 :	0	0	0	0	0	0	0	SX8	
Registre 34 :	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	ordonnée source (0 à 1023)
Registre 35 :	0	0	0	0	0	0	SY9	SY8	

SX0 à SX8 = Abscisse du pixel d'origine.

SY0 à SY9 = Ordonnée du pixel d'origine.

Registre 40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	déplacement horizontal (0 à 511)
Registre 41	0	0	0	0	0	0	0	NX8	
Registre 42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	déplacement vertical (0 à 1023)
Registre 43	0	0	0	0	0	0	NY9	NY8	

NX0 à NX8 = Nombre de pixels à placer dans la direction horizontale

NY0 à NY9 = Nombre de pixels à placer dans la direction verticale.

Registre 44

CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
-----	-----	-----	-----	-----	-----	-----	-----

 donnée

CR0 à CR7 = Couleur.

En SCREEN 5 et 7, les CR4 à CR7 bits sont ignorés.

En SCREEN 6, les CR2 à CR7 bits sont ignorés.

Registre 45

0	0	MXD	0	DIY	DIX	0	0
---	---	-----	---	-----	-----	---	---

 paramètres

DIX = Direction horizontale pour NX. 0 pour droite ; 1 pour gauche.

DIY = Direction verticale pour NY. 0 pour bas ; 1 pour haut.

MXD = Mémoire vidéo de destination. 0 pour sélectionner la VRAM principale ; 1 pour accéder à la mémoire vidéo étendue sur les MSX2 avec 192 Ko de VRAM.

- Écrire le code de la commande sur les 4 bits de poids fort ainsi que le code de l'opérateur logique désiré sur les 4 bits de poids faible du registre 46 pour exécuter la commande.

Registre 46 :

1	0	1	1	LO3	LO2	LO1	LO0
---	---	---	---	-----	-----	-----	-----

 opérateur

LO0 à LO3 = Opérateur logique à appliquer.

0000 = IMP

1000 = TIMP

0001 = AND

1001 = TAND

0010 = OR

1010 = TOR

0011 = XOR

1011 = TXOR

0100 = NOT

1100 = TNOT

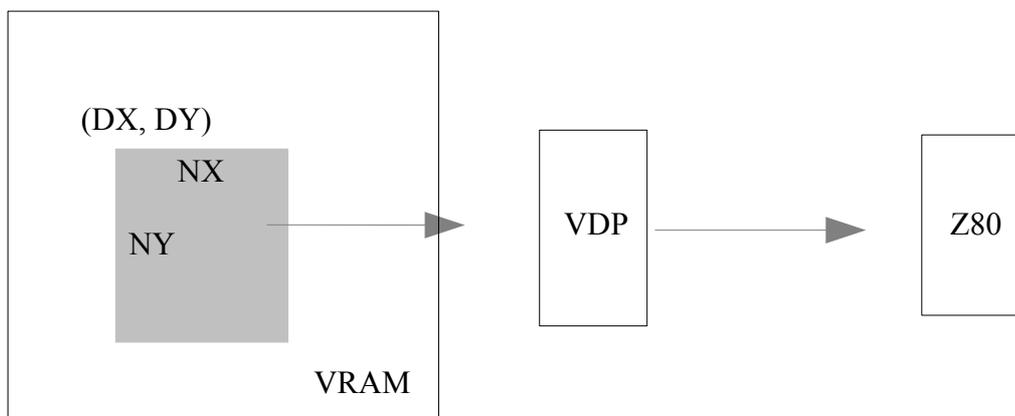
- Lire le registre de statut 2.
- Tester l'état du bit CE. Si celui-ci est à 0, alors le processus n'est pas terminé, sinon on passe à l'étape suivante.
- Tester l'état du bit TR pour savoir si le transfert s'est effectué, si celui-ci se trouve à 0, alors le VDP n'est pas prêt à recevoir l'octet suivant, recommencer en 3. Si par contre ce bit est à 1, effectuer l'étape suivante.
- Envoyer l'octet suivant à mettre en VRAM dans le registre 44 (le premier octet a été traité à l'étape 1) puis reprendre l'opération à l'étape 3.

Une fois l'instruction terminée, les registres de commande du VDP se trouveront dans l'état suivant :

32~33	34~35	36~37	38~39	40~41	42~43	44	45
		idem	modifié	idem	modifié	idem	idem

LMCM (Logical Move to CPU from Memory)

Schéma :



Fonction : LMCM permet de récupérer en mémoire central (via le Z80) le contenu d'une zone de la mémoire vidéo. Le transfert se fait pixel par pixel.

Source : VRAM

Destination : Z80

Voici la marche à suivre pour exécuter l'instruction LMCM :

1. Écrire les paramètres dans les registres suivants.

	bit 7	bit 6	bit 5	bit 4	bit 4	bit 2	bit 1	bit 0	
Registre 32 :	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	abscisse source (0 à 511)
Registre 33 :	0	0	0	0	0	0	0	SX8	
Registre 34 :	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	ordonnée source (0 à 1023)
Registre 35 :	0	0	0	0	0	0	SY9	SY8	

SX0 à SX8 = Abscisse du pixel d'origine.

SY0 à SY9 = Ordonnée du pixel d'origine.

Registre 40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	déplacement horizontal (0 à 511)
Registre 41	0	0	0	0	0	0	0	NX8	
Registre 42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	déplacement vertical (0 à 1023)
Registre 43	0	0	0	0	0	0	NY9	NY8	

NX0 à NX8 = Nombre de pixels à transférer dans la direction horizontale

NY0 à NY9 = Nombre de pixels à transférer dans la direction verticale.

Registre 45	0	0	0	MXS	DIY	DIX	0	0	paramètres
-------------	---	---	---	-----	-----	-----	---	---	------------

DIX = Direction horizontale pour NX. 0 = Droite ; 1 = Gauche.

DIY = Direction verticale pour NY. 0 = Bas ; 1 = Haut.

MXS = 0 pour sélectionner la VRAM principale.

1 pour accéder à la mémoire vidéo étendue sur les MSX2 avec 192 Ko de VRAM.

2. Lire le registre de statut 7 afin de mettre le bit TR du registre de statut 2 à 0.
3. Écrire 10100000B (0A0h) dans le registre 46 pour exécuter la commande.
4. Lire le registre de statut 2.
5. Tester l'état du bit TR, si celui-ci se trouve à 0, c'est que le VDP n'a pas encore transféré la donnée, continuer à l'étape 6. Si par contre, ce bit est à 1, lire le registre de statut 7. Il contient la donnée issue de la mémoire vidéo.
6. Tester l'état du bit CE, si celui-ci est à 0, alors le processus est terminée. Dans le cas contraire (CE à 1), on recommence à l'étape 4.

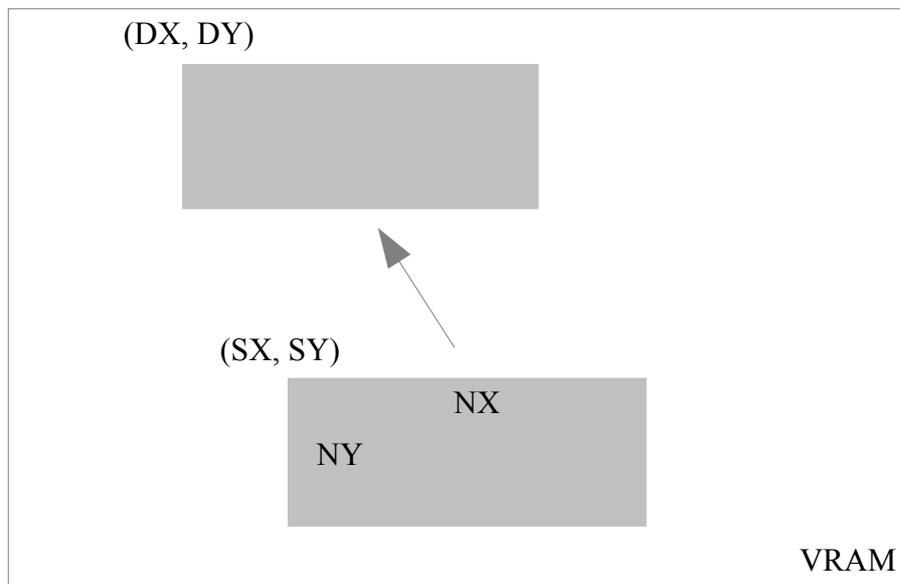
Note : Après la lecture de la dernière donnée, même si TR est à 1, la commande sera terminée lorsque le bit CE sera à 0.

Une fois l'instruction terminée, les registres de commande du VDP se trouveront dans l'état suivant :

32~33	34~35	36~37	38~39	40~41	42~43	44	45
idem	modifié			idem	modifié	code	idem

LMMM (Logical Move to Memory from Memory)

Schéma :



Fonction : LMMM fait la copie d'une zone de mémoire vidéo vers une autre une autre zone. La copie se fait pixel par pixel.

Source : VRAM.

Destination : VRAM.

Voici la marche à suivre pour exécuter l'instruction LMMM:

1. Écrire les paramètres dans les registres suivants.

	bit 7	bit 6	bit 5	bit 4	bit 4	bit 2	bit 1	bit 0	
Registre 32 :	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	abscisse source (0 à 511)
Registre 33 :	0	0	0	0	0	0	0	SX8	
Registre 34 :	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	ordonnée source (0 à 1023)
Registre 35 :	0	0	0	0	0	0	SY9	SY8	

SX0 à SX8 = Abscisse du pixel d'origine.

SY0 à SY9 = Ordonnée du pixel d'origine.

Registre 36 :	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	abscisse destination (0 à 511)
Registre 37 :	0	0	0	0	0	0	0	DX8	
Registre 38 :	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	ordonnée destination (0 à 1023)
Registre 39 :	0	0	0	0	0	0	DY9	DY8	

DX0 à DX8 = Abscisse du premier pixel de destination.

DY0 à DY9 = Ordonnée du pixel de destination.

Registre 40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	déplacement horizontal (0 à 511)
Registre 41	0	0	0	0	0	0	0	NX8	
Registre 42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	déplacement vertical (0 à 1023)
Registre 43	0	0	0	0	0	0	NY9	NY8	

NX0 à NX8 = Nombre de pixels à copier dans la direction horizontale

NY0 à NY9 = Nombre de pixels à copier dans la direction verticale.

Registre 45	0	0	MXD	MXS	DIY	DIX	0	0	paramètres
-------------	---	---	-----	-----	-----	-----	---	---	------------

DIX = Direction horizontale pour NX. 0 pour droite ; 1 pour gauche.

DIY = Direction verticale pour NY. 0 pour bas ; 1 pour haut.

MXS = Mémoire vidéo source. 0 pour sélectionner la VRAM principale ; 1 pour accéder à la mémoire vidéo étendue sur les MSX2 avec 192 Ko de VRAM.

MXD = Mémoire vidéo de destination. 0 pour sélectionner la VRAM principale ; 1 pour accéder à la mémoire vidéo étendue sur les MSX2 avec 192 Ko de VRAM.

- Écrire le code de la commande sur les 4 bits de poids fort ainsi que le code de l'opérateur logique désiré sur les 4 bits de poids faible du registre 46 pour exécuter la commande.

Registre 46 :	1	0	0	1	LO3	LO2	LO1	LO0	opérateur
---------------	---	---	---	---	-----	-----	-----	-----	-----------

LO0 à LO3 = Opérateur logique à appliquer.

0000 = IMP

1000 = TIMP

0001 = AND

1001 = TAND

0010 = OR

1010 = TOR

0011 = XOR

1011 = TXOR

0100 = NOT

1100 = TNOT

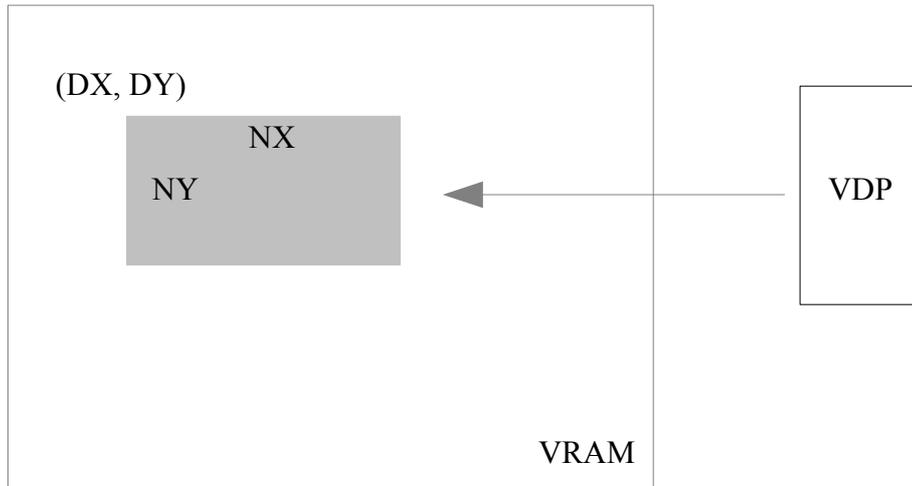
- Le bit CE du registre de statut 2 restera à 1 tant que le processus n'est pas terminé. Vérifiez que ce bit soit à 0 avant d'exécuter la commande suivante ou de changer de paramètre.

Une fois l'instruction terminée, les registres du processeur vidéo se trouveront dans l'état suivant :

32~33	34~35	36~37	38~39	40~41	42~43	44	45
idem	modifié	idem	modifié	idem	modifié		idem

LMMV (Logical Move to Memory from VDP)

Schéma :



Fonction : LMMV permet de remplir la mémoire vidéo avec une seule donnée. A la différence des routines du Bios (BIGFIL ou FILVRM), la mémoire vidéo est définie par des coordonnées en X et Y. Le remplissage se fait pixel par pixel, ce qui autorise l'emploi d'un opérateur logique.

Source : VDP.

Destination : VRAM.

Voici la marche à suivre pour exécuter l'instruction LMMV:

1. Écrire les paramètres dans les registres suivants.

	bit 7	bit 6	bit 5	bit 4	bit 4	bit 2	bit 1	bit 0	
Registre 32 :	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	abscisse source (0 à 511)
Registre 33 :	0	0	0	0	0	0	0	SX8	
Registre 34 :	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	ordonnée source (0 à 1023)
Registre 35 :	0	0	0	0	0	0	SY9	SY8	

SX0 à SX8 = Abscisse du pixel d'origine.

SY0 à SY9 = Ordonnée du pixel d'origine.

Registre 40	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0	déplacement horizontal (0 à 511)
Registre 41	0	0	0	0	0	0	0	NX8	
Registre 42	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0	déplacement vertical (0 à 1023)
Registre 43	0	0	0	0	0	0	NY9	NY8	

NX0 à NX8 = Nombre de pixels à placer dans la direction horizontale

NY0 à NY9 = Nombre de pixels à placer dans la direction verticale.

Registre 44

CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
-----	-----	-----	-----	-----	-----	-----	-----

 donnée

CR0 à CR7 = Couleur.

En SCREEN 5 et 7, les CR4 à CR7 bits sont ignorés.

En SCREEN 6, les CR2 à CR7 bits sont ignorés.

R45

0	0	MXD	0	DIY	DIX	0	0
---	---	-----	---	-----	-----	---	---

 paramètres

DIX = Direction horizontale pour NX. 0 pour droite, 1 pour gauche.

DIY = Direction verticale pour NY. 0 pour bas, 1 pour haut.

MXD = 0 pour sélectionner la VRAM principale.

1 pour accéder à la mémoire vidéo étendue sur les MSX2 avec 192 Ko de VRAM.

- Écrire le code de la commande sur les 4 bits de poids fort ainsi que le code de l'opérateur logique désiré sur les 4 bits de poids faible du registre 46 pour exécuter la commande.

Registre 46 :

1	0	0	0	LO3	LO2	LO1	LO0
---	---	---	---	-----	-----	-----	-----

 opérateur

LO0 à LO3 = Opérateur logique à appliquer.

0000 = IMP

1000 = TIMP

0001 = AND

1001 = TAND

0010 = OR

1010 = TOR

0011 = XOR

1011 = TXOR

0100 = NOT

1100 = TNOT

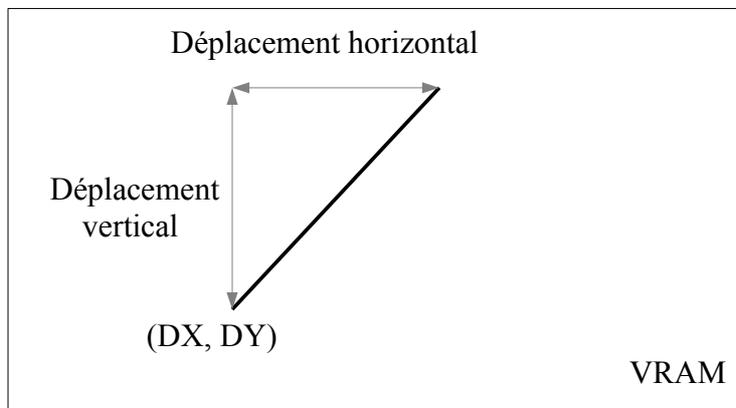
- Le bit CE du registre de statut 2 restera à 1 tant que le processus n'est pas terminé. Vérifiez que ce bit soit à 0 avant d'exécuter la commande suivante ou de changer de paramètre.

Une fois l'instruction terminée, les registres du processeur vidéo se trouveront dans l'état suivant :

32~33	34~35	36~37	38~39	40~41	42~43	44	45
		idem	modifié	idem	modifié	idem	idem

LINE (draw a LINE)

Schéma :



Fonction : LINE trace une ligne à l'écran. L'utilisateur définit un triangle rectangle et le processeur vidéo en trace l'hypoténuse.

Source : VDP

Destination : VRAM

Voici la marche à suivre pour exécuter l'instruction LMMV :

1. Écrire les paramètres dans les registres suivants.

	bit 7	bit 6	bit 5	bit 4	bit 4	bit 2	bit 1	bit 0	
Registre 36 :	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0	abscisse destination (0 à 511)
Registre 37 :	0	0	0	0	0	0	0	DX8	
Registre 38 :	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0	ordonnée destination (0 à 1023)
Registre 39 :	0	0	0	0	0	0	DY9	DY8	

DX0 à DX8 = Abscisse du premier pixel du tracé.

DY0 à DY9 = Ordonnée du premier pixel du tracé.

Registre 40 :	MJ7	MJ6	MJ5	MJ4	MJ3	MJ2	MJ1	MJ0	déplacement le plus long (0 à 1023)
Registre 41 :	0	0	0	0	0	0	MJ9	MJ8	
Registre 42 :	MI7	MI6	MI5	MI4	MI3	MI2	MI1	MI0	déplacement le plus court (0 à 511)
Registre 43 :	0	0	0	0	0	0	MI9	MI8	

NX0 à NX8 = Déplacement le plus long.

NY0 à NY9 = Déplacement le plus court.

Registre 44 :	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	donnée
---------------	-----	-----	-----	-----	-----	-----	-----	-----	--------

CR0 à CR7 = Couleur du tracé.

En SCREEN 5 et 7, les CR4 à CR7 bits sont ignorés.

En SCREEN 6, les CR2 à CR7 bits sont ignorés.

Registre 45 :

0	0	MXD	0	DIY	DIX	0	MAJ
---	---	-----	---	-----	-----	---	-----

 paramètres

MAJ = Mettre à 1 pour que le déplacement le plus long se fasse à la verticale.

DIX = Mettre à 1 pour tracer la ligne vers la gauche.

DIY = Mettre à 1 pour tracer la ligne vers le haut.

MXD = Mémoire vidéo de destination. 0 pour la VRAM principale ; 1 pour accéder à la mémoire vidéo étendue sur les MSX2 avec 192 Ko de VRAM.

2. Écrire le code de la commande sur les 4 bits de poids fort ainsi que le code de l'opérateur logique désiré sur les 4 bits de poids faible du registre 46 pour exécuter la commande.

Registre 46 :

0	1	1	1	LO3	LO2	LO1	LO0
---	---	---	---	-----	-----	-----	-----

 Opérateur

LO0 à LO3 = Opérateur logique à appliquer.

0000 = IMP	1000 = TIMP
0001 = AND	1001 = TAND
0010 = OR	1010 = TOR
0011 = XOR	1011 = TXOR
0100 = NOT	1100 = TNOT

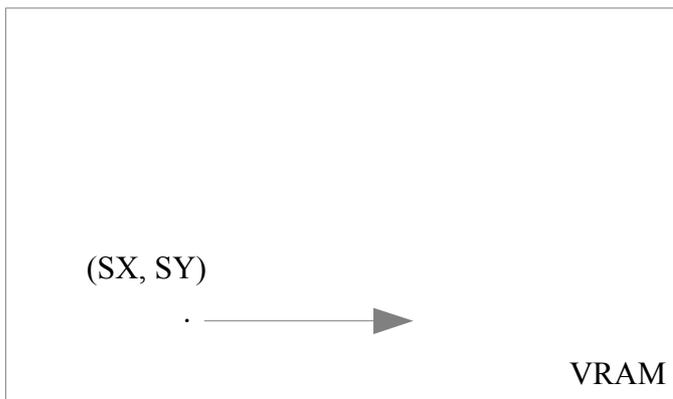
3. Le bit CE du registre de statut 2 restera à 1 tant que le processus n'est pas terminé. Vérifier que ce bit soit à 0 avant d'exécuter la commande suivante ou de changer de paramètre.

Une fois l'instruction terminée, les registres de commande du VDP se trouveront dans l'état suivant :

32~33	34~35	36~37	38~39	40~41	42~43	44	45
		idem	modifié	idem	idem	idem	idem

SRCH (SeaRCH for color)

Schéma :



Fonction : SRCH permet de rechercher une couleur donnée sur une ligne horizontale. Cette instruction est particulièrement utile lors du remplissage d'une surface (PAINT).

Source : VDP.

Destination : VRAM.

Voici la marche à suivre pour exécuter l'instruction SRCH :

1. Écrire les paramètres dans les registres suivants.

	bit 7	bit 6	bit 5	bit 4	bit 4	bit 2	bit 1	bit 0	
Registre 32 :	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	abscisse source (0 à 511)
Registre 33 :	0	0	0	0	0	0	0	SX8	
Registre 34 :	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	ordonnée source (0 à 1023)
Registre 35 :	0	0	0	0	0	0	SY9	SY8	

SX0 à SX8 = Abscisse du pixel d'origine.

SY0 à SY9 = Ordonnée du pixel d'origine.

Registre 44	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	donnée
-------------	-----	-----	-----	-----	-----	-----	-----	-----	--------

CR0 à CR7 = Couleur à trouver..

En SCREEN 5 et 7, les CR4 à CR7 bits sont ignorés.

En SCREEN 6, les CR2 à CR7 bits sont ignorés.

R45	0	0	MXD	0	0	DIX	EQ	0	paramètres
-----	---	---	-----	---	---	-----	----	---	------------

EQ = 0 pour stopper la recherche lorsque la couleur est trouvée ; 1 pour stopper la recherche lorsque la couleur est différente de celle paramétrée.

DIX = Direction horizontale pour le tracé à partir du pixel origine.

0 pour tracer la ligne vers la droite ; 1 pour tracer la ligne vers la gauche.

MXD = Mémoire vidéo de destination. 0 pour la VRAM principale ; 1 pour accéder à la mémoire vidéo étendue sur les MSX2 avec 192 Ko de VRAM.

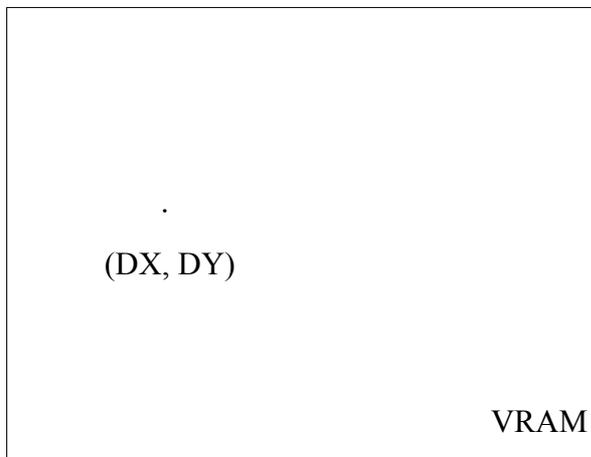
2. Écrire l'octet 01100000B (060h) dans le registre 46 exécuter la commande.
3. Lire le registre de statut 2.
4. Tester l'état du bit CE, si celui-ci est à 1, on retourne à l'étape 3. Si le bit CE est à 0, on continue.
5. Tester l'état du bit BD. Si celui-ci se trouve à 0, alors le processeur vidéo n'a pas trouvé de pixel de la couleur recherchée, la commande est terminée. Si, au contraire, BD est à 1, le VDP a bien repéré le pixel de couleur cherchée. L'abscisse de ce pixel s'obtient en lisant le contenu des registres d'état 8 et 9. Le premier renferme les 8 bits de poids faible de cette abscisse, alors que le bit 0 du registre 9 donne le bit de poids fort.

Une fois l'instruction terminée, les registres du processeur vidéo se trouveront dans l'état suivant :

32~33	34~35	36~37	38~39	40~41	42~43	44	45
idem	idem					idem	idem

PSET (Point SET)

Schéma :



Fonction : PSET affiche un point dans la mémoire vidéo

Source : VDP.

Destination : VRAM.

Voici la marche à suivre pour exécuter l'instruction PSET :

1. Écrire les paramètres dans les registres suivants.

	bit 7	bit 6	bit 5	bit 4	bit 4	bit 2	bit 1	bit 0	
Registre 32 :	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	abscisse source (0 à 511)
Registre 33 :	0	0	0	0	0	0	0	SX8	
Registre 34 :	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	ordonnée source (0 à 1023)
Registre 35 :	0	0	0	0	0	0	SY9	SY8	

SX0 à SX8 = Abscisse du pixel d'origine.

SY0 à SY9 = Ordonnée du pixel d'origine.

R44	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	couleur
-----	-----	-----	-----	-----	-----	-----	-----	-----	---------

CR0 à CR7 = Couleur du point à placer.

En SCREEN 5 et 7, les CR4 à CR7 bits sont ignorés.

En SCREEN 6, les CR2 à CR7 bits sont ignorés.

R45	0	0	MXD	0	0	0	0	0	paramètres
-----	---	---	-----	---	---	---	---	---	------------

MXD = 0 pour sélectionner la VRAM principale.

1 pour accéder à la mémoire vidéo étendue sur les MSX2 avec 192 Ko de VRAM.

2. Écrire le code de la commande sur les 4 bits de poids fort ainsi que le code de l'opérateur logique désiré sur les 4 bits de poids faible du registre 46 pour exécuter la commande.

Registre 46 :

0	1	0	1	LO3	LO2	LO1	LO0
---	---	---	---	-----	-----	-----	-----

 opérateur

LO0 à LO3 = Opérateur logique à appliquer.

0000 = IMP	1000 = TIMP
0001 = AND	1001 = TAND
0010 = OR	1010 = TOR
0011 = XOR	1011 = TXOR
0100 = NOT	1100 = TNOT

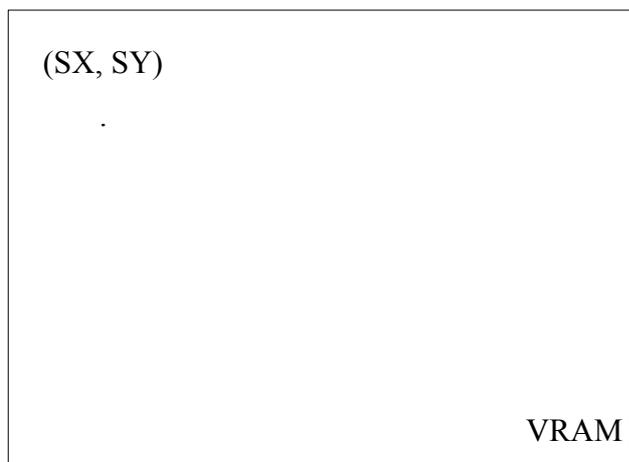
3. Le bit CE du registre de statut 2 restera à 1 tant que le processus n'est pas terminé. Vérifier que ce bit soit à 0 avant d'exécuter la commande suivante ou de changer de paramètre.

Une fois l'instruction terminée, les registres de commande du VDP se trouveront dans l'état suivant :

32~33	34~35	36~37	38~39	40~41	42~43	44	45
		idem	idem			idem	idem

INSTRUCTION POINT (is POINT set ?)

Schéma :



Fonction : POINT donne la couleur d'un pixel dans la mémoire vidéo.

Source : VRAM

Destination : VDP.

Voici la marche à suivre pour exécuter l'instruction POINT :

1. Écrire les paramètres dans les registres suivants.

	bit 7	bit 6	bit 5	bit 4	bit 4	bit 2	bit 1	bit 0	
Registre 32 :	SX7	SX6	SX5	SX4	SX3	SX2	SX1	SX0	abscisse source (0 à 511)
Registre 33 :	0	0	0	0	0	0	0	SX8	
Registre 34 :	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	ordonnée source (0 à 1023)
Registre 35 :	0	0	0	0	0	0	SY9	SY8	

SX0 à SX8 = Abscisse du pixel d'origine.

SY0 à SY9 = Ordonnée du pixel d'origine.

Registre 45 :	0	0	0	MXS	0	0	0	0	paramètres
---------------	---	---	---	-----	---	---	---	---	------------

MXS = Mémoire vidéo source. 0 pour sélectionner la VRAM principale ; 1 pour accéder à la mémoire vidéo étendue sur les MSX2 avec 192 Ko de VRAM.

2. Écrire le code de la commande sur les 4 bits de poids fort ainsi que le code de l'opérateur logique désiré sur les 4 bits de poids faible du registre 46 pour exécuter la commande.

Registre 46 :	0	1	0	0	LO3	LO2	LO1	LO0	opérateur
---------------	---	---	---	---	-----	-----	-----	-----	-----------

LO0 à LO3 = Opérateur logique à appliquer.

0000 = IMP

1000 = TIMP

0001 = AND

1001 = TAND

0010 = OR	1010 = TOR
0011 = XOR	1011 = TXOR
0100 = NOT	1100 = TNOT

3. Lire le registre de statut 2, tester l'état du bit CE. Si celui-ci est à 1, alors l'instruction n'est pas terminée, recommencer l'étape 3. Lorsque le bit CE passe à 1, on peut poursuivre avec l'étape 4.
4. Lire le registre de statut 7, il contient le code de la couleur du pixel.

Une fois l'instruction terminée, les registres de commande du VDP se trouveront dans l'état suivant :

32~33	34~35	36~37	38~39	40~41	42~43	44	45
idem	idem					code	idem

6.9 Les différents modes d'affichage (SCREEN 0 à SCREEN 12)

Nous allons voir exactement comment fonctionne chaque mode d'affichage.

SCREEN 0 (WIDTH 40) (MSX1~)

Résolution : 256 × 192 pixels.
 Type : Mode texte, 40 colonnes × 24 lignes, caractères 6 × 8.
 Couleurs : 2 couleurs parmi 15. (2 parmi 512 à partir de MSX2.)
 Sprites : *Inutilisés.*
 Taille d'une page écran : 4 Ko.
 Nombre de pages maximum : 32 avec 128 Ko de VRAM.

Table des caractères :	00000h~003BFH	960 octets.
Table des couleurs des caractères :	<i>Inutilisée.</i>	
Table des formes des caractères :	00800h~00FFFH	2048 octets.
Table des attributs des Sprites :	<i>Inutilisée.</i>	
Table des formes des Sprites :	<i>Inutilisée.</i>	
Table des couleurs des Sprites :	<i>Inutilisée.</i>	
Table de la palette (MSX2~) :	00400h~0041FH	32 octets.

Description du fonctionnement du mode texte :

Exécutons le programme Basic suivant.

```

10 SCREEN0: WIDTH40: CLS
20 PRINT"HELLO!"

```

Le texte « HELLO! » s'affiche sur le coin tout en haut à gauche de l'écran. Ces caractères sont disposés en VRAM de la façon suivante dans la table des caractères :

Adresse	+0	+1	+2	+3	+4	+5	+6	...
00000h	48h (H)	45h (E)	4Ch (L)	4Ch (L)	4Fh (O)	21h (!)	20h	...
00028h	20h	20h	20h	20h	20h	...		
00050h	20h	20h	20h	...				
00078h	20h					
⋮								

Chaque caractères ont une taille de 6x8 mais prennent une place de 8x8 dans la table des formes des caractères. À l'initialisation du SCREEN 0, la table des formes est importée depuis la Main-ROM et disposées dans l'ordre du code ASCII.

Table des formes des caractères :

Caractère 00h	Caractère 01h	Caractère 02h	Caractère 03h	... jusqu'au FFh
00800h	00808h	00810h	00818h	
00801h	00809h	00811h	00819h	
00802h	0080Ah	00812h	0081Ah	
00803h	0080Bh	00813h	0081Bh	
00804h	0080Ch	00814h	⋮	
00805h	0080Dh	00815h		
00806h	0080Eh	00816h		
00807h	0080Fh	00817h		

Écran :

Ligne 0	H	E	L	L	O	!		...
Ligne 1								...
Ligne 2				...				
⋮								

La position de chaque caractère à l'écran est codé sur un octet dans la table des caractères. Il y a 40 colonnes multipliées par 24 lignes, soit 960 caractères donc 960 octets. Chaque octet renferme le code ASCII du caractère à afficher à l'écran. Ainsi, si à l'écran se trouvent uniquement les caractères « HELLO » dans le coin supérieur gauche, la table des caractères commencera par :

00000h	72 (048h), code du « H »
00001h	69 (045h), code du « E »
00002h	76 (04Ch), code du « L »
00003h	76 (04Ch), code du « L »
00004h	79 (04Fh), code du « O »
00005h	33 (021h), code du « ! »
00006h	32 (045h), code de l'espace

Le reste de la table serait alors rempli de 32 (020h), code de l'espace (ou éventuellement de 0), si il n'y a pas d'autre texte à l'écran.

La table des formes des caractères contient, elle, 2048 octets regroupés par paquets de 8 octets. Ce qui donne 2048 divisé par 8, soit 256 paquets. Chaque paquet correspond à un caractère (code ASCII) et en définit la forme. Par exemple le 65ème paquet code le caractère « A » et ressemble à ceci :

Le caractère n°65, donc l'octet 65*8 = 520 (208h). On trouvera donc l'information qui nous intéresse à l'adresse de début de la table des caractères (00800h) plus 208h, soit 00A08h.

			7	6	5	4	3	2	1	0										
00A08h	contient	32	=	020h	=	00100000B	=>			X										
00A09h	contient	80	=	050h	=	01010000B	=>		X			X								
00A0Ah	contient	136	=	088h	=	10001000B	=>	X					X							
00A0Bh	contient	136	=	088h	=	10001000B	=>	X					X							
00A0Ch	contient	248	=	0F8h	=	11111000B	=>	X	X	X	X	X								
00A0Dh	contient	136	=	088h	=	10001000B	=>	X					X							
00A0Eh	contient	136	=	088h	=	10001000B	=>	X					X							
00A0Fh	contient	0	=	000h	=	00000000B	=>													

Les deux bits de poids faible doivent toujours rester à 0 et ne sont de toute manière jamais pris en compte (car tout caractère se trouve défini dans une matrice 6 × 8).

Redéfinissons à présent le caractère « A », faisons la démarche en sens inverse.

7	6	5	4	3	2	1	0		
X	X	X	X	X	X			=>	11111100B = 0FCh
X		X			X			=>	10100100B = 0A4h
X					X			=>	10000100B = 084h
X		X			X			=>	10100100B = 0A4h
X		X			X			=>	10100100B = 0A4h
X			X		X			=>	10010100B = 094h
X	X			X	X			=>	11001100B = 0CCh
X		X	X		X			=>	10110100B = 0B4h

soit les valeurs
en hexadécimal

Pour cela, il suffit d'exécuter le petit programme suivant :

```
10 SCREEN 0:WIDTH 40:CLS:AD=&HA08
20 FOR I=0 TO 7:READ A$:VPOKE AD+I,VAL("&H"+A$):NEXT I
30 PRINT"A A A A"
50 DATA FC,A4,84,A4,A4,94,CC,B4
```

Tous les « A » de l'écran se transforment en signes cabalistiques d'un ésotérisme certain. Un simple SCREEN 0 fera revenir les choses à la normale.

SCREEN 0 (WIDTH 80) (MSX2~)

Résolution :	512 × 192 / 212 pixels.
Type :	Mode texte, 80 colonnes × 24 / 26,5 lignes, caractères 6 × 8.
Couleurs :	4 couleurs parmi 512.
Sprites :	<i>Inutilisés.</i>
Taille d'une page écran :	8 Ko.
Nombre de pages maximum :	16 avec 128 Ko de VRAM.

Table des caractères (24 lignes) :	00000h~0077Fh	1920 octets
" (26,5 lignes) :	00000h~0086Fh	2160 octets
Table des couleurs (24 lignes) :	00800h~008EFh	240 octets
" (26,5 lignes) :	00800h~0090Dh	270 octets
Table des formes des caractères :	01000h~017FFh	2048 octets
Table des attributs des Sprites :	<i>Inutilisée</i>	
Table des formes des Sprites :	<i>Inutilisée</i>	
Table des couleurs des Sprites :	<i>Inutilisée</i>	
Table de la palette (MSX2~) :	00F00h~00F1Fh	32 octets

Redéfinissons à présent le caractère « B », faisons la démarche en sens inverse

7	6	5	4	3	2	1	0		en binaire	en hexadécimal
X	X	X	X	X				=>	11111000B	= 0F8h
X				X				=>	10001000B	= 088h
X		X	X	X				=>	10111000B	= 0B8h
X			X	X				=>	10011000B	= 098h
X		X	X	X				=>	10111000B	= 0B8h
X				X				=>	10001000B	= 088h
X				X				=>	10001000B	= 088h
X	X	X	X	X				=>	11111000B	= 0F8h

Il suffit à présent d'exécuter le petit programme suivant :

```
10 SCREEN 0: WIDTH 80: CLS: AD=&H1210
20 FOR I=0 TO 7:READ A$:VPOKE AD+I,VAL("&H"+A$):NEXT I
30 PRINT"B B B B"
50 DATA F8,88,B8,98,B8,88,88,F8,00
```

Tous les « B » de l'écran se transforment en petit signe « E » en vidéo inverse. L'instruction SCREEN 0 fera revenir les choses à la normale.

La table des couleurs permet de définir la couleur des caractères lors d'un clignotement le texte. Chaque bit correspond à un caractère à l'écran, voir le paragraphe sur [les registres de contrôle du processeur vidéo](#) (Registres 12 et 13).

SCREEN 1 (MSX1~)

- Résolution : 256 × 192 pixels.
- Type : Mode texte, 32 colonnes × 24 lignes, caractères 8 × 8.
- Couleurs : 16 couleurs. (16 couleurs parmi 512 à partir du MSX2).
- Sprites : Type 1.
- Taille d'une page écran : 4 Ko.
- Nombre de pages maximum : 32 avec 128 Ko de VRAM.

Table des caractères :	01800h~01AFFh	768 octets.
Table des couleurs :	02000h~0201Fh	32 octets.
Table des formes des caractères :	00000h~007FFh	2048 octets.
Table des attributs des Sprites :	01B00h~01B7Fh	128 octets.
Table des formes des Sprites :	03800h~03FFFh	2048 octets.
Table des couleurs des Sprites :	<i>Inutilisée.</i>	
Table de la palette (MSX2~) :	02020h~0203Fh	32 octets.

Le SCREEN 1 fonctionne de manière proche du mode texte 40 colonnes (voir ci-dessus pour plus de précisions). La table des caractères est toujours constituée d'un octet par caractère à l'écran, soit cette fois 32 colonnes fois 24 lignes donc 768 octets. Chaque octet renferme le code ASCII du caractère à afficher à l'écran. Ainsi, si à l'écran se trouvent uniquement les caractères « A BEN » dans le coin supérieur gauche, la table des caractères commencera par :

01800H	65 (41h), code du « A »
01801H	32 (020h), code de l'espace
01802H	66 (042h), code du « B »
01803H	69 (045h), code du « E »
01804H	78 (04Eh), code du « N »
01805H	32 (020h), code de l'espace
⋮	⋮

Le reste de la table serait alors rempli de 32 (020h), code de l'espace (ou éventuellement de 0).

La table des formes des caractères contient aussi 2048 octets regroupés par paquets de 8 octets. Ce qui donne 2048 divisé par 8, soit 256 paquets. Chaque paquet correspond à un caractère (code ASCII) et en définit la forme. Par exemple, le 32ème paquet code le caractère d'espacement. Le paquet n°32, donc octet $32 \times 8 = 256$ (100h). On trouvera donc l'information qui nous intéresse à l'adresse de début de la table des formes (00000h) plus 100h, soit 0100h. Les cases de 00100h à 00107h en VRAM sont toutes remplies de 0 (Espace = rien). Un simple VPOKE &H107, 255 devrait vous convaincre de l'utilité de bien connaître le processeur vidéo.

Notez que dans ce mode, tous les bits peuvent être utilisés pour définir un caractère (voir les modes 40 et 80 colonnes) car les caractères se trouvent cette fois définis dans une matrice 8 sur 8).

Essayez donc le programme Basic suivant (tout à fait désopilant) :

```
10 SCREEN 1:WIDTH 32
20 LOCATE 12,12:PRINT"COUCOU"
30 FOR J=0 TO 7
40 FOR I=0 TO 7
50 VPOKE &H100+I,2^J
60 NEXT I,J
70 GOTO 30
```

Ce que l'on fait avec 7 lignes Basic, c'est fou ! Après un BREAK, l'instruction SCREEN 1 fera revenir les choses à la (triste) normale.

Quant à la table des couleurs, rien de bien affriolant dans ce mode. Les 32 octets de la table des couleurs codent chacun la couleur d'une suite de huit caractères du code ASCII. Ce qui signifie par exemple que le contenu de 02009h donne la couleur des lettres H à O. Les 4 bits de poids fort codent la couleur du texte alors que les 4 bits de poids faible codent la couleur du fond. Le seul moyen d'utiliser un tel système est de redéfinir le jeu de caractères? A part cela, on peut toujours s'amuser à changer la couleur du curseur (code 255 ou 0FFh) :

```
10 SCREEN 1: WIDTH 32: COLOR 10,0
```

20 VPOKE &H201F, &HD7

L'exécution de ce programme rend le curseur mauve (0Dh) avec des caractères cyan (07h).

Après toutes ces émotions, et suivant l'adage « All work and no play makes Jack a sad boy », je vous propose un petit jeu. Profitez-en, amusez-vous bien, et hop (fonctionne sur MSX1 en retirant les instructions COLOR =) :

```
10 SCREEN 1: WIDTH 32: COLOR 10,1: Sprite$(1)="I"+CHR$(127)+"I"+CHR$(8)+
"I"+CHR$(127)+"I"+CHR$(8): PRINT: PRINT: FOR I=1 TO 4:PRINT"- RALLY -";:
NEXT: LOCATE 5,12: INPUT"NIVEAU (1 ou 2)";NN: NN=NN+1:IF NN=3 THEN N=4: DD=4
ELSE N=8: DD=2
20 FOR I=0 TO 7: VPOKE&H208+I,128: NEXT: FOR I=0 TO 7:VPOKE&H210+I,1: NEXT:
T=3:X=40:Y=88: PUT Sprite 1, (X,Y), 9:COLOR 10,2: FOR I=0 TO 23: PRINT TAB(T-
1);"B"+STRING$(N+1,"O")+"A": NEXT: VPOKE&H2004, &HC2: VPOKE&H2009,17:
VPOKE&H2006, &HF2: COLOR=(2,1,1,2)
30 LOCATE 20,5:PRINT"Attention": FOR J=3 TO 1 STEP -1:LOCATE 23,7: PRINT J :
PLAY"L8D": FOR I=1 TO 800: NEXT I,J: LOCATE 20,5: PRINT SPACE$(9):LOCATE
24,7:PRINT" ": LOCATE 0,24:PLAY"L2O6A"
40 SC=SC+1: PUT Sprite 1, (X,Y), 8:I=&H1960+X/8: IF VPEEK(I)<>79 OR VPEEK
(I+1)<>79 THEN 100 ELSE D=INT(RND(14)*3):D=-D*(T>2 AND T+N<29)-2*(T<=2)-
(T+N>=29): ON D GOTO 60,70
50 PRINT TAB (T-1);"B"+STRING$(N+1,"O")+"A":GOTO 60
60 T=T-1:PRINT TAB (T);"/"+STRING$(N,"O")+"/":GOTO 80
70 T=T+1:PRINT TAB (T-1);"\ "+STRING$(N,"O")+"\ ":GOTO 80
80 I=STICK(0): IF I=3 THEN X=X+DD else IF I=7 THEN X=X-DD
90 GOTO 40
100 VPOKE&H2007, &HF2: VPOKE&H200A, &H72: VPOKE&H200C, &H72: VPOKE&H200D, &H72:
VPOKE&H200E, &H72: LOCATE 1,23: PRINT"Score =";SC*10;: FOR I=1 TO 100000!:
NEXT
```

Voici l'exemple parfait de ce que l'on peut faire pour sa petite voisine par un après-midi pluvieux, gris.

SCREEN 2

(MSX1~/MSX2~)

Résolution : 256 × 192 pixels.
Type : Caractères graphiques 8 × 8, 32 colonnes sur 3 × 8 lignes.
Couleurs : 15 couleurs dont 2 par ligne de caractère. (MSX1)
(16 couleurs parmi 512 au lieu de 15 à partir du MSX2)
Sprites : Type 1.
Taille d'une page écran : 16 Ko.
Nombre de pages maximum : 8 avec 128 Ko de VRAM.

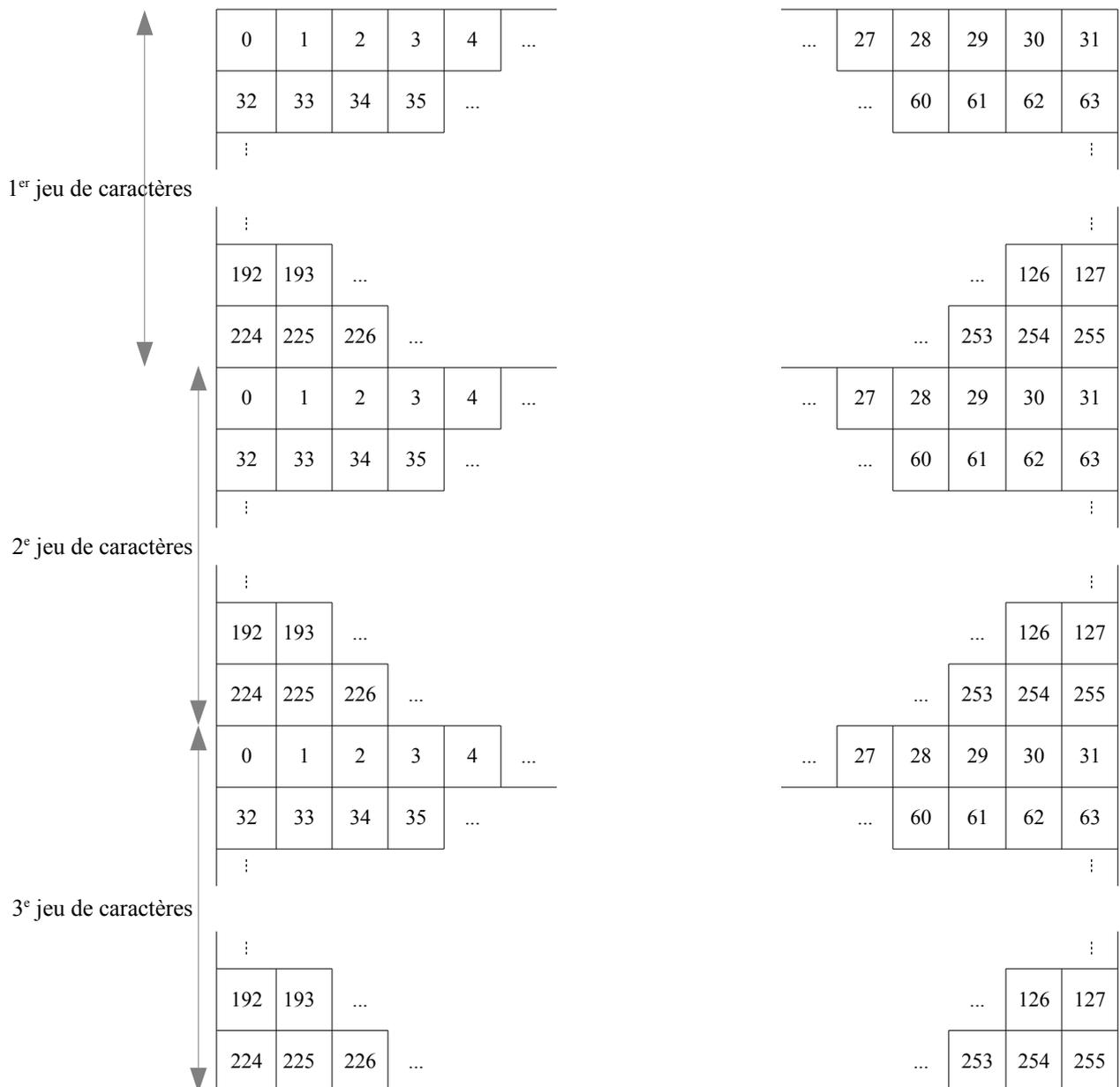
Table des caractères :	01800h~01AFFh	768 octets
Table des couleurs :	02000h~037FFh	6144 octets
Table des formes :	00000h~017FFh	6144 octets
Table des attributs des Sprites :	01B00h~01B7Fh	128 octets
Table des formes des Sprites :	03800h~03FFFh	2048 octets

Table des couleurs des Sprites : *Inutilisée*

Table de la palette (MSX2~) : 02020h~0203Fh 32 octets

Le SCREEN 2 ressemble beaucoup à un mode texte dans son utilisation mais dans ce mode, tous les caractères sont graphiques. Ils ne sont pas définis selon le code ASCII. L'écran est divisé en 3 bandes horizontales. Chaque bande est composée d'un jeu de 256 caractères (de 0 à 255 par défaut).

Format de la table des caractères :



La table des formes a le même format qu'en SCREEN1. Elle contient 2048 octets regroupés par paquets de 8 octets. Ce qui donne 2048 divisé par 8, soit 256 paquets. Chaque paquet correspond à un caractère. Sachant que dans ce mode d'écran, il y a 3 jeux de caractères, la table des formes fait donc 3×2048 octets. Les bits de ces octets à 1 représentent les points du tracé. Ceux à 0 représentent le fond du caractère graphique.

La table des couleurs définit la couleur pour chaque ligne d'un caractère. Les 4 bits de poids fort codent la couleur du tracé (0 ~ 15), alors que les 4 bits de poids faible codent la couleur de fond (0 ~ 15). Par exemple si l'adresse 00009h contient 081h alors que 02009h renferme 0D1h, alors les pixels de coordonnées (8, 1) et (15, 1) seront allumés en magenta (0Dh) alors que les pixels (9, 1) à (14, 1) resteront (ou deviendront) noirs.

Bien entendu, rien n'empêche le programmeur averti de modifier la table des caractères. Il suffit de faire attention à quel tiers d'écran on désire accéder. Voyons un petit exemple en Basic :

```
10 SCREEN 2: COLOR 10,1,1: CLS
20 '
30 ' TABLE DES FORMES, FORME N°10
40 FOR I=0 TO 7: READ A$: VPOKE I,VAL("&H"+A$): NEXT
50 '
60 ' TABLE DES COULEURS, FORME N°0
70 FOR I=&H2000 TO &H2007: READ A$: VPOKE I,VAL("&H"+A$): NEXT
80 '
90 ' TABLE DES MOTIFS, JOLI CADRE EN BRIQUES
100 FOR I=&H1800 TO &H181F: VPOKE I,0: NEXT
110 FOR I=&H18E0 TO &H18FF: VPOKE I,0: NEXT
120 FOR I=&H1800 TO &H18E0 STEP 32 :VPOKE I,0: NEXT
130 FOR I=&H181F TO &H18FF STEP 32 :VPOKE I,0: NEXT
140 '
150 CIRCLE(127,98),50,14: PAINT(127,98),14
160'
170 GOTO 170
180 '
190 ' DONNÉES POUR LA FORME
200 DATA 00,BB,00,DD,00,BB,00,DD
210 '
220 ' DONNÉES POUR LES COULEURS
230 DATA 00,60,00,50,00,A0,00,20
```

Nous avons redéfini la forme 0, puis nous l'avons affichée sur tout la première et la septième ligne, ainsi que sur les côtés. Essayez donc de faire la même chose avec le Basic classique.

Vous remarquerez, à la ligne 150, que tous les dessins s'effectuent derrière notre cadre (ce qui est logique mais agréable). Attention cependant à ne pas utiliser les pixels dans le cadre entre (0, 0) et (7, 7).

SCREEN 3

MSX1/MSX2

Résolution :	64 × 48.
Type :	Caractères graphiques 2 × 4 pixels.
Couleurs :	15 couleurs (MSX1) 16 couleurs parmi 512 (MSX2~)
Sprites :	type 1
Taille d'une page écran :	4 Ko
Nombre de pages maximum :	32 avec 128 Ko de VRAM

Table des caractères :	00800h~00AFFh	768 octets
Table des couleurs :		
Table des formes :	00000h~007FFh	2048 octets
Table des attributs des Sprites :	01B00h~01B7Fh	128 octets
Table des formes des Sprites :	03800h~03FFFh	2048 octets
Table des couleurs des Sprites :	<i>Inutilisée</i>	
Table de la palette (MSX2~) :	02020h~0203Fh	32 octets

Le mode d'écran 3 fonctionne comme le mode 2 (voir le SCREEN 2 pour plus de précisions), mais avec un écran de 64 sur 48. En matière de taille, un pixel mode 3 équivaut à 4 pixels mode 2.

0, 0	4, 0	8, 0		4 « gros » pixels
0, 3	7, 3		15, 3	
0, 4	4, 4	8, 4		4 « gros » pixels
0, 7	7, 7		15, 7	

Dans la table des formes, chaque octet code 2 gros pixels. Les 4 bits de poids fort donnent la couleur (0 à 15) du pixel d'abscisse paire, alors que les 4 bits de poids faible indiquent la couleur (0 à 15) du pixel d'abscisse impaire.

Si l'on néglige la table Bitmap, on ne travaille que sur la table des formes qui ressemble à :

00000H	0, 0	63, 0
00100H	0, 7	63, 7
	0, 8	63, 8
00200H	0, 15	63, 15
	0, 16	63, 16
00300H	0, 23	63, 23
	0, 24	63, 24
00400H	0, 31	63, 31
	0, 32	63, 32
00500H	0, 39	63, 39
	0, 40	63, 40
	0, 47	63, 47

Il est alors facile d'adresser n'importe quel pixel, comme le montre l'exemple suivant en Basic :

```

10 SCREEN 3 : COLOR 10, 1, 1 : CLS
20 '
30 X = 25 : Y = 117 : C = 6 : PSET (X, Y), 7 : FOR I = 1 TO 1000 : NEXT
40 VPOKE (INT(Y/32) * 256) + INT(X/8)*8 + (Y MOD 32)/4, -(C*16)*(X MOD 4 < 2) - C*(X MOD
4 > 1)
140 GOTO 140

```

Pour ceux qui désirent profiter des possibilités offertes par la table des caractères, sachez que chaque octet de cette table contient le numéro de la forme à utiliser. Seulement, on n'utilise que deux lignes de la forme suivant la règle :

```

lignes 0 et 1 --- VRAM de 00800h à 0081Fh
lignes 2 et 3 --- VRAM de 00820h à 0083Fh
lignes 4 et 5 --- VRAM de 00840h à 0085Fh
lignes 6 et 7 --- VRAM de 00860h à 0087Fh
lignes 0 et 1 --- VRAM de 00880h à 0089Fh
lignes 2 et 3 --- VRAM de 008A0h à 008BFh
et ainsi de suite ...

```

Voici un nouvel exemple Basic utilisant cette technique :

```

10 SCREEN 3: COLOR 10,1,1: CLS
20 '
30 ' TABLE DES FORMES, FORME N°0
40 VPOKE 0,&HA8: VPOKE 1,&H8A: VPOKE 2,&HA8: VPOKE 3,&H8A: VPOKE 4,&HA8:
VPOKE 5,&H8A: VPOKE 6,&HA8: VPOKE 7,&H8A
50 '
60 ' TABLE DES CARACTÈRES, JOLI CADRE EN BRIQUES
70 FOR I=0 TO 31 : VPOKE &H800+I,0: NEXT
80 FOR I=0 TO 31 : VPOKE &HA00+I,0: NEXT
90 FOR I=&H800 TO &HA00 STEP 32: VPOKE I,0: NEXT
100 FOR I=&H81F TO &HA1F STEP 32: VPOKE I,0: NEXT
110 '
120 CIRCLE(127,98),50,14: PAINT(127,98),14
130 '
140 GOTO 140

```

Notez qu'en SCREEN 3, lorsque vous utilisez les instructions Basic OPEN « GRP: » AS #1 puis PRINT #1, « Texte » le texte s'affiche en grandes lettres (4 fois la taille normale 40 colonnes).

SCREEN 4

MSX2

Résolution :	256 × 192 avec contraintes de couleur
Texte :	8 × 8 / 32 caractères sur 24 lignes
Couleurs :	16 couleurs parmi 512
Sprites :	Type 2
Taille d'une page écran :	16 Ko
Nombre de pages maximum :	8 avec 128 Ko de VRAM

Table des caractères :	01800h~01AFFh	768 octets
Table des couleurs :	02000h~037FFh	6144 octets
Table des formes :	00000h~017FFh	6144 octets
Table des attributs Sprite :	01E00h~01E7Fh	128 octets
Table des formes de Sprite :	03800h~03FFFh	2048 octets
Table des couleurs de Sprite :	01C00h~0D7FFh	512 octets
Table de la palette des couleurs :	01E80h~01E9Fh	32 octets

Le SCREEN 4 est en tous points identique au SCREEN 2 à l'exception des Sprites qui sont de type 2. Voir le paragraphe concernant le SCREEN 2 pour plus d'information sur l'affichage. Pour de plus amples renseignements sur les Sprites, voir le paragraphe 6.10 « [Les Sprites et leur fonctionnement](#) ».

SCREEN 5 MSX2

Résolution :	256 × 212 / 192
Type :	Bitmap
Couleurs :	16 couleurs parmi 512
Sprites :	Type 2
Taille d'une page écran :	32 Ko
Nombre de pages maximum :	4 avec 128 Ko de VRAM

Table Bitmap de 192 lignes :	00000h~05FFFh	24576 octets
" 212 lignes :	00000h~069FFh	27136 octets
Table des couleurs :		
Table des formes :		
Table des attributs Sprite :	07600h~0767Fh	128 octets
Table des formes de Sprite :	07800h~07FFFh	2048 octets
Table des couleurs de Sprite :	07400h~075FFh	512 octets
Table de la palette des couleurs :	07680h~0769Fh	32 octets

Avec le mode d'écran 5, on entre dans le domaine des graphismes en « bitmap ». Ce terme signifie que chaque pixel est codé directement par sa couleur, et ce, indépendamment des autres pixels. Il n'existe donc plus de contrainte de couleur, ni même de table de couleur ou de formes. Ce mode « bitmap » apparaît donc comme idéal puisqu'il allie puissance et simplicité.

Dans ce mode, chaque octet de la table Bitmap code deux pixels. Les 4 bits de poids fort donnent la couleur (0 ~ 15) du pixel d'abscisse paire, alors que les 4 bits de poids faible donnent la couleur (0 ~ 15) du pixel d'abscisse impaire.

Le registre 2 du VDP indique la page à afficher (0 à 3) :

Registre 2 :	0	A16	A15	1	1	1	1	1
--------------	---	-----	-----	---	---	---	---	---

A16 et A15 : page à afficher 0 à 3.

SCREEN 6

MSX2

Résolution : 512 × 212 / 192
 Type : Bitmap
 Couleurs : 4 couleurs parmi 512
 Sprites : type 2
 Taille d'une page écran : 32 Ko
 Nombre de pages maximum : 4 avec 128 Ko de VRAM

Table Bitmap de 192 lignes :	00000h~05FFFh	24576 octets
" 212 lignes :	00000h~069FFh	27136 octets
Table des couleurs :	<i>inutilisée</i>	
Table des formes :	<i>inutilisée</i>	
Table des attributs Sprite :	07600h~0767Fh	128 octets
Table des formes de Sprite :	07800h~07FFFh	2048 octets
Table des couleurs de Sprite :	07400h~075FFh	512 octets
Table de la palette des couleurs :	07680h~0769Fh	32 octets

En SCREEN 6, chaque octet de la table Bitmap code 4 pixels. Les bits 6 et 7 (poids fort) définissent la couleur (0 ~ 3) du pixel dont l'abscisse est multiple de 4. Les bits 4 et 5 codent la couleur du pixel immédiatement à la droite de celui dont l'abscisse est multiple de 4, et ainsi de suite...

La table Bitmap ressemble donc à :

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
00000h	pixel (0,0)	pixel (1,0)	pixel (2,0)	pixel (3,0)				
00001h	pixel (4,0)	pixel (5,0)	pixel (6,0)	pixel (7,0)				
⋮								
0007Fh	pixel (508,0)	pixel (509,0)	pixel (510,0)	pixel (511,0)				
00080h	pixel (0,1)	pixel (1,1)	pixel (2,1)	pixel (3,1)				
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Le processeur vidéo ne peut afficher que 4 couleurs en SCREEN 6. C'est vrai, mais tout le monde sait

qu'en juxtaposant deux couleurs, avec des pixels suffisamment petits, on obtient une troisième couleur. Les concepteurs du VDP ont conçu une instruction qui mélange automatiquement deux couleurs (« hardware tiling »). Cette fonction peut s'appliquer à la marge (ainsi, 7 couleurs de marge sont possibles) et surtout aux Sprites de la manière suivante :

- pour la marge, préciser la couleur comme suit :

0	0	0	1	X	X	Y	Y	
				!	!	!	!	
				!	!	+-----+--	+-----+--	Couleur des pixels abscisse impaire
				+-----+	+-----+	Couleur des pixels abscisse paire		

- pour les Sprites, préciser la couleur comme suit :

0	0	0	0	X	X	Y	Y	
				!	!	!	!	
				!	!	+-----+-----	+-----+-----	Couleur partie gauche du pixel
				+-----+	+-----+	Couleur partie droite du pixel		

Voici un exemple en Basic :

```

10 VDP(9)=VDP(9)OR &H20: ' Pour utiliser la couleur 0
20 SCREEN 6: COLOR 10,0,&B11001: CLS
30 COLOR=(1,7,0,0) : COLOR=(2,0,7,0) : COLOR=(3,0,0,7)
40 FOR I=10 TO 310 STEP 100: LINE(I,100)-(I+75,200),(I-10)/100,BF: NEXT
50 Sprite$(1)=STRING$(8,255)
60 OPEN"GRP:" AS #1: PRESET(110,80):PRINT#1,"SCREEN 6 : QUATRE COULEURS MAXI
!"
70 FOR I=0 TO 248: PUT Sprite 1,(I,25),&B0111: NEXT
80 FOR I=248 TO 0 STEP -1: PUT Sprite 1,(I,25),&B0111: NEXT
90 GOTO 70

```

Le registre 2 du V9938 indique la page à afficher (0 à 3) :

Registre 2 :	0	A16	A15	1	1	1	1	1
--------------	---	-----	-----	---	---	---	---	---

A16 et A15 = Page à afficher 0 à 3.

Les modes d'écran 7 et 8 ne sont disponibles que sur les MSX2 équipés de 128 Ko de mémoire vive vidéo (VRAM).

SCREEN 7

MSX2

Résolution :	512 × 192 / 212
Type :	Bitmap
Couleurs :	16 couleurs parmi 512
Sprites :	Type 2
Taille d'une page écran :	64 Ko
Nombre de pages maximum :	2 avec 128 Ko de VRAM

Table Bitmap de 192 lignes :	00000h~0BFFFh	49152 octets
------------------------------	---------------	--------------

"	212 lignes :	00000h~0D3FFh	54272 octets
Table des couleurs :		<i>inutilisée</i>	
Table des formes :		<i>inutilisée</i>	
Table des attributs Sprite :		0FA00h~0FA7Fh	128 octets
Table des formes de Sprite :		0F000h~0F7FFh	2048 octets
Table des couleurs de Sprite :		0F800h~0F9FFh	512 octets
Table de la palette des couleurs :		0FA80h~0FA9Fh	32 octets

Dans ce mode, chaque octet de la table Bitmap code deux pixels. Les 4 bits de poids fort donnent la couleur (0 ~ 15) du pixel d'abscisse paire, alors que les 4 bits de poids faible donnent la couleur (0 ~ 15) du pixel d'abscisse impaire.

Le registre 2 du VDP indique la page à afficher (0 à 3) :

Registre 2 :	0	0	A16	1	1	1	1	1
--------------	---	---	-----	---	---	---	---	---

A16 et A15 = Page à afficher 0 à 1.

SCREEN 8

MSX2

Résolution :	256 × 192 / 212 bitmap
Type :	Bitmap
Couleurs :	256 couleurs
Sprites :	Type 2
Taille d'une page écran :	64 Ko
Nombre de pages maximum :	2 avec 128 Ko de VRAM

Table Bitmap de 192 lignes :	00000h~0BFFFh	49152 octets	
"	212 lignes :	00000h~0D3FFh	54271 octets
Table des couleurs :		<i>inutilisée</i>	
Table des formes :		<i>inutilisée</i>	
Table des attributs Sprite :	0FA00h~0FA7Fh	128 octets	
Table des formes de Sprite :	0F000h~0F7FFh	2048 octets	
Table des couleurs de Sprite :	0F800h~0F9FFh	512 octets	
Table de la palette des couleurs :	0FA80h~0FA9Fh	32 octets	

Dans ce mode, chaque octet de la table des Bitmap définit la couleur d'un pixel (0-255).

La couleur est déterminée de la manière suivante :

Octet :

G2	G1	G0	R2	R1	R0	B1	B0
----	----	----	----	----	----	----	----

G0 à G2 = Intensité de vert (0 ~ 7).

R0 à R2 = Intensité de rouge (0 ~ 7).

B0 et B1 = Intensité de bleu (0 ~ 3).

En Basic, on peut calculer le numéro de la couleur désirée avec la formule :

$$100 C = 32 * G + 4 * R + B$$

Le registre 2 du VDP indique la page à afficher (0 à 3) :

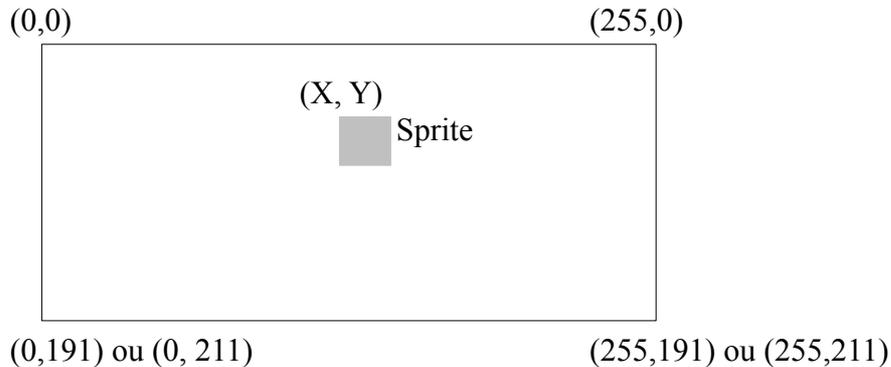
Registre 2 :

0	0	A16	A15	1	1	1	1
---	---	-----	-----	---	---	---	---

A16 et A15 = Page à afficher 0 à 1.

6.10 Les Sprites et leur fonctionnement

Les Sprites, ou lutins graphiques, sont des motifs graphiques pouvant être affichés automatiquement à n'importe quel pixel dans n'importe quelle couleur, ceci en se superposant aux dessins dans l'écran, sans les effacer. Ces facilités les prédestinent au mouvement puisqu'il suffit de modifier deux octets (les coordonnées du Sprite) pour faire déplacer tout un motif sans toucher au reste de l'écran. Le VDP peut mémoriser jusqu'à 256 Sprites, il peut en afficher simultanément au maximum 32 à l'écran. Les Sprites se trouvent définis dans un matrice 8 sur 8 ou 16 sur 16. Ils peuvent être de 2 tailles, normale ou double.



Attention : En SCREEN 6 et 7, les coordonnées du côté droit de l'écran n'est pas (511,Y) mais bien (255,Y). Les Sprites se déplacent de 2 pixel par 2. Pour vous en persuader, essayez donc ce petit programme...

```
10 SCREEN 7: COLOR 10,0,0
20 LINE(0,0)-(511,0),15
30 Sprite$(0)=CHR$(&HFF)
40 FOR I=0 TO 255
50 PUT Sprite 0,(I,0),8
60 NEXT
70 GOTO 40
```

... puis celui-ci :

```
10 SCREEN 7: COLOR 10,0,0
20 LINE(0,0)-(511,0),15
30 Sprite$(0)=CHR$(&HFF)
40 FOR I=0 TO 255
50 PUT Sprite 0,(I,255),8: ' ici Y=255
60 NEXT
70 GOTO 40
```

Il existe deux modes de fonctionnement des Sprites bien distincts. Le VDP choisit automatiquement le mode adéquat suivant le mode écran. En SCREEN 1, 2 et 3, c'est le mode Sprites type 1 appelé aussi mode restreint (qui correspond au MSX1) alors qu'en SCREEN 4 à 8, c'est le mode Sprites type 2 ou mode étendu (MSX2).

Le mode Sprites type 1

(MSX1/MSX2)

Fonctionnement :

Dans ce mode, il peut y avoir 32 Sprites à l'écran numérotés de 0 à 31. Plus un Sprite a un numéro faible, plus il est prioritaire. Ceci signifie que si le Sprite n°3 croise le Sprite n°16, on verra le Sprite n°3 passer dessus (lorsque des pixels des deux Sprites se trouveront aux mêmes coordonnées, c'est les pixels du Sprite n°3 qui seront affichés).

4 Sprites peuvent, au maximum, se trouver sur la même ligne. A partir du 5ème Sprite, toutes les portions communes aux 5 Sprites disparaissent sur le (ou les) Sprites de plus faible priorité.

Paramètres :

Matrice des Sprites Le bit 1 (SI) du registre 1 du VDP définit la taille de la matrice.

SI à 0 pour Sprites de 8×8 pixels.

SI à 1 pour Sprites de 16×16 pixels.

Taille des Sprites Le bit 0 (MAG) du registre 1 du VDP permet de doubler la taille.

MAG à 0 pour taille des pixels normale.

MAG à 1 pour taille des pixels double.

Pour bien comprendre le mécanisme de la taille, voici un exemple en Basic :

```
10 SCREEN 2: COLOR 10,0,0: CLS
20 OPEN"GRP:" AS #1
30 Sprite$(0)="ABABABAB"
40 PRESET(16,5): PRINT #1,"Un Sprite Quelconque.": COLOR
8,1: PRESET(16,155): PRINT #1,"MAG ="
50 LINE(63,155)-(72,164),1,BF: PRESET(64,155): PRINT #1,"0"
60 VDP(1)=VDP(1) AND &HFE
70 FOR I=0 TO 248
80 PUT Sprite 0,(I,35),7
90 NEXT
100 LINE(63,155)-(72,164),1,BF: PRESET(64,155): PRINT #1,"1"
110 VDP(1)=VDP(1) OR 1
120 FOR I=0 TO 248
130 PUT Sprite 0,(I,35),7
140 NEXT
150 GOTO 50
```

Collisions :

Lorsque deux Sprites entrent en collision (deux pixels ou plus avec les mêmes coordonnées), le bit 5 du registre de statut 0 passe à 1.

5 Sprites :

Lorsque plus de 4 Sprites se trouvent sur la même ligne, le bit 6 (5S) du registre de statut 0 passe à 1. De plus les 5 bits de poids faible du registre de statut 0 donnent le numéro du 5ème Sprite.

Utilisation :

Pour afficher un Sprite, il faut d'abord le définir dans la table des formes de Sprites.

Pour les Sprites 8 sur 8, il suffit de 8 octets pour définir un Sprite, sachant qu'un bit représente un pixel,

par exemple, en SCREEN 2, la table des formes de Sprites pourrait être :

		huit octets pour le Sprite n°0 qui aura cette forme								
		7	~					0		
03800h	11111111	X	X	X	X	X	X	X	X	X
03801h	10100001	X		X						X
03802h	10010001	X			X					X
03803h	10001001	X				X				X
03804h	10000101	X					X		X	
03805h	10000101	X					X		X	
03806h	10001001	X				X				X
03807h	11111111	X	X	X	X	X	X	X	X	X

Dans le cas de Sprites 16 sur 16, les choses deviennent un tout petit peu plus compliquées. Le Sprite se décompose en 4 parties :

0	2
1	3

abscisses entre 247 et 255 (pour un Sprite 8 sur 8). Par contre, pour le faire entrer, il faut utiliser le bit EC (voir ci-dessous).

Le numéro de Sprite donne la forme à utiliser à l'affichage. On peut définir 256 Sprites 8 sur 8, ou 64 Sprites 16 sur 16. Pour les Sprites 8 sur 8, on choisit simplement le numéro de celui que l'on désire afficher. Pour les Sprites 16 sur 16, on donne le numéro de Sprite multiplié par quatre.

La couleur est toujours codée sur 4 bits.

Le bit EC (« Early Clock ») sert à décaler le Sprite de 32 pixels vers la gauche. Ceci permet de faire « entrer » un Sprite à l'écran comme dans l'exemple suivant :

```
10 SCREEN 2: COLOR 10,0,0: CLS
20 Sprite$(1)=STRING$(8,255)
30 FOR I=24 TO 40: FOR J=0 TO 100: NEXT J
40 PUT Sprite 0,(I,35): VPOKE &H1B03,&H87
50 NEXT I
55 IF NOT (STRIG(0)) THEN 55
```

Le mode 2 (MSX2~)

Fonctionnement :

Dans ce mode, il peut y avoir 32 Sprites à l'écran numérotés de 0 à 31. Plus un Sprite a un numéro faible, plus il est prioritaire. Ceci signifie que si le Sprite n°3 croise le Sprite n°16, on verra le Sprite n°3 passer dessus (lorsque des pixels des deux Sprites se trouveront aux mêmes coordonnées, c'est les pixels du Sprite n°3 qui seront affichés).

8 Sprites peuvent, au maximum, se trouver sur la même ligne. A partir du 9ème Sprite, toutes les portions communes aux 9 Sprites disparaissent sur le (ou les) Sprites de plus faible priorité.

Paramètres :

Matrice des Sprites Le bit 1 (SI) du registre 1 du VDP définit la taille de la matrice.

SI à 0 pour Sprites de 8 × 8 pixels.

SI à 1 pour Sprites de 16 × 16 pixels.

Taille des Sprites Le bit 0 (MAG) du registre 1 du VDP permet de doubler la taille.

MAG à 0 pour taille des pixels normale.

MAG à 1 pour taille des pixels double.

État des Sprites Le bit 1 (SPD) du registre 8 du VDP permet de désactiver des Sprites.

SPD à 0 = Les Sprites s'affichent. (Sprites désactivés.)

SPD à 1 = Les Sprites ne s'affichent pas. (Sprites activés.)

Collisions :

Lorsque deux Sprites entrent en collision (deux pixels ou plus ont les mêmes coordonnées), le bit 5 du registre de statut passe à 1 (si le bit CC est à 0). Le bit 5 est automatiquement remis à 0 lors d'une lecture du registre de statut 2. Les registres de statut n°3, 4, 5 et 6 indiquent alors les coordonnées du pixel où la collision s'est produite (à condition que les bits MS et LP du registre n°8 du processeur soient tous les deux à 0).

Registre de statut 3 :

X7	X6	X5	X4	X3	X2	X1	X0
----	----	----	----	----	----	----	----

Registre de statut 4 :

1	1	1	1	1	1	1	X8
---	---	---	---	---	---	---	----

Registre de statut 5 :

Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
----	----	----	----	----	----	----	----

Registre de statut 6 :

1	1	1	1	1	1	Y9	Y8
---	---	---	---	---	---	----	----

Lorsque le registre n°5 est lu, les registres sont tous les 4 remis à 0.

Une possibilité supplémentaire est offerte au programmeur dans le mode 2. En effet, si le bit CC est mis à 1 dans la table des couleurs de Sprite, alors un « ou » logique (OR) est effectué avec la même ligne du Sprite suivant dans la table des formes lorsqu'il est superposées.

9 Sprites :

Lorsque plus de 8 Sprites se trouvent sur la même ligne, le bit 6 (5S) du registre de statut 0 passe à 1. De plus les 5 bits de poids faible du registre de statut 0 donnent le numéro du 9ème Sprite (0 à 31).

Utilisation :

Pour afficher un Sprite, il faut d'abord le définir dans la table des formes de Sprites.

Pour les Sprites 8 sur 8, il suffit de 8 octets pour définir un Sprite, sachant qu'un bit représente un pixel, par exemple, en SCREEN 7, la table des formes de Sprites pourrait être :

		huit octets pour le Sprite n°0 qui aura cette forme										
		7	~	0								
0F000H	11111111	X	X	X	X	X	X	X	X	X	X	X
0F001H	11000011	X	X							X	X	
0F002H	10100101	X		X				X			X	
0F003H	10011001	X				X	X				X	
0F004H	10011001	X				X	X				X	
0F005H	10100101	X		X				X			X	
0F006H	11000011	X	X							X	X	
0F007H	11111111	X	X	X	X	X	X	X	X	X	X	X

Dans le cas de Sprites 16 sur 16, les choses deviennent un tout petit peu plus compliquées. Le Sprite se décompose en 4 parties :

0	2
1	3

	7 ~ 0	7 6 5 4 3 2 1 0	
0F000H	11111110	X X X X X X X	
0F001H	10000010	X	X
0F002H	10001110	X	X X X
0F003H	10001000	X	X
0F004H	10001000	X	X
0F005H	10001000	X	X
0F006H	10001000	X	X
0F007H	10001000	X	X
0F008H	10001000	X	X
0F009H	10001000	X	X
0F00AH	10001000	X	X
0F00BH	10001000	X	X
0F00CH	10001000	X	X
0F00DH	10001111	X	X X X X
0F00EH	10000000	X	
0F00FH	11111111	X X X X X X X X	
0F010H	01111111	X X X X X X X	
0F011H	01000001	X	X
0F012H	01110001	X X X	X
0F013H	00010001	X	X
0F014H	00010001	X	X
0F015H	00010001	X	X
0F016H	00010001	X	X
0F017H	00010001	X	X
0F018H	00010001	X	X
0F019H	00010001	X	X
0F01AH	00010001	X	X
0F01BH	00010001	X	X
0F01CH	00010001	X	X
0F01DH	11110001	X X X X	X
0F01EH	00000001		X
0F01FH	11111111	X X X X X X X X	

huit octets pour le premier quart du Sprite n°0

huit octets pour le second quart du Sprite n°0

huit octets pour le troisième quart du Sprite n°0

huit octets pour le dernier quart du Sprite n°0

La réunion des quatre parties donne le résultat ci-dessous :

couleur	vert			rouge			bleu			
	R2	R1	R0	G2	G1	G0	B2	B1	B0	
0000B	0	0	0	0	0	0	0	0	0	(Noir)
0001B	0	0	0	0	0	0	0	1	0	(Bleu foncé)
0010B	0	0	0	0	1	1	0	0	0	(Rouge foncé)
0011B	0	0	0	0	1	1	0	1	0	(Violet foncé)
0100B	0	1	1	0	0	0	0	0	0	(Vert foncé)
0101B	0	1	1	0	0	0	0	1	0	(Turquoise)
0110B	0	1	1	0	1	1	0	0	0	(Olive)
0111B	0	1	1	0	1	1	0	1	0	(Gris)
1000B	1	0	0	1	1	1	0	1	0	(Orange clair)
1001B	0	0	0	0	0	0	1	1	1	(Bleu)
1010B	0	0	0	1	1	1	0	0	0	(Rouge)
1011B	0	0	0	1	1	1	1	1	1	(Violet)
1100B	1	1	1	0	0	0	0	0	0	(Vert)
1101B	1	1	1	0	0	0	1	1	1	(Bleu clair)
1110B	1	1	1	1	1	1	0	0	0	(Jaune)
1111B	1	1	1	1	1	1	1	1	1	(Blanc)

Notez que vous pouvez régler la couleur, aussi que les autres bits, pour CHAQUE ligne du Sprite. Notez également qu'on utilise toujours 16 octets, même pour les Sprites 8 sur 8.

Une fois le Sprite défini, on peut le manipuler à souhait grâce à la table des attributs de Sprites.

Cette table fonctionne avec des plans graphiques, et compte 4 octets par plan. On appelle « plan graphique » un espace comprenant un élément graphique. Il y a donc 32 plans (numérotés de 0 à 31) avec chacun un Sprite, puis un plan avec les graphismes ordinaires, enfin parfois un plan en entrée vidéo.

La table des attributs de Sprites code les caractéristiques de chaque plan sur 4 octets :

	7	6	5	4	3	2	1	0	
01B00H	ordonnée Sprite plan 0 (0-255)								} attributs du plan 0
01B01H	abscisse Sprite plan 0 (0-255)								
01B02H	numéro de Sprite (0-255)								
01B03H	---- réservé au système ----								
01B04H	ordonnée Sprite plan 1 (0-255)								}
⋮	⋮								

L'ordonnée peut varier entre 0 et 255. Si le Sprite sort de l'écran, le VDP n'affichera que la partie visible du Sprite. En réalité, on utilise les coordonnées de 225 (-31 en complément à 2) à 255, et 0 à 191. Ceci permet de faire entrer un Sprite dans l'écran (-31 à -1 ou 225 à 255) ou au contraire de le faire sortir (183 à 191 pour un 8 sur 8) petit à petit.

Notez que si l'ordonnée de n'importe quel Sprite est à 216 (0D8h), alors tous les Sprites dans les plans supérieurs deviennent invisibles. Par exemple si l'on met l'ordonnée du Sprite contenu dans 8 à 216, alors tous les Sprites dans les plans 9 à 31 disparaissent.

L'abscisse prend une valeur entre 0 et 255. Dans les modes à 512 pixels de large (SCREEN 6 et 7), un pixel Sprite équivaut à deux pixels graphiques. Pour faire sortir le Sprite on utilise les abscisses entre 247 et 255 (pour un Sprite 8 sur 8). Par contre, pour le faire entrer, il faut utiliser le bit EC dans la table des couleurs.

Le numéro de Sprite donne la forme à utiliser à l'affichage. On peut définir 256 Sprites 8 sur 8, ou 64 Sprites 16 sur 16. pour les Sprites 8 sur 8, on choisit simplement le numéro de celui que l'on désire afficher. Pour les Sprites 16 sur 16, on donne le numéro de Sprite multiplié par quatre.

6.11 La souris et le crayon optique

Bien que le V9938 est capable de gérer la souris et le crayon optique, ces deux périphériques de pointage ne sont pas être gérés par le processeur vidéo sur MSX. Ces fonctions ont d'ailleurs été retirées du V9958. Il est nécessaire d'utiliser le Bios pour cela (voir le chapitre 3, la routine GTPAD (000DBh en Main-ROM) ou NEWPAD (001ADh en Sub-ROM)).

7 Des applications types

Ce chapitre est destiné à soulager le programmeur en lui proposant des solutions « prêtes à l'emploi » aux problèmes qu'on rencontre dans la majorité des applications. Muni de cette « bibliothèque de base », le programmeur ne perdra plus de temps en recherches inutiles et pourra se consacrer entièrement à la programmation de son application.

7.1 Revenir à l'interpréteur Basic

Il est souvent utile de pouvoir retourner à l'interpréteur Basic depuis un programme en langage machine, autrement que par le RET du Z80. Pour cela il faut :

1. Sélectionner la Main-ROM en pages 0 et 1.
2. Effectuer un saut en 0409Bh.

Ce qui donnerait, en assembleur, les lignes de programme suivantes :

```
                ORG ADRS
;
MNROM           EQU 0FCC1H
ENASLT          EQU 00024H
RETURN          EQU 0409BH
;
                .....
;
ADRS            LD A, (MNROM)
                LD HL,0           ; Bios en page 0
                CALL ENASLT
                LD HL,04000H      ; Basic en page 1
                CALL ENASLT
                JP RETURN
```

Lorsque l'on travaille sous Basic, ou que l'on ne touche pas aux Slot, il suffit d'effectuer le JP 0409Bh.

7.2 Quel type de MSX ?

Pour savoir si l'ordinateur utilisé est un MSX1 ou un MSX2, il suffit d'examiner le contenu de la case mémoire 0002Dh en Main-ROM.

contenu de 02DH	ordinateur
0	MSX1
1	MSX2
2	MSX2+
3	MSX turbo R
4 à 255	non défini

Attention : La Main-ROM ne se trouve pas toujours dans un Slot primaire. Il faut lire le contenu de la

variable système MNROM (0FCC1h) pour savoir si le Slot de la Main-ROM se trouve sur un Slot secondaire ou non.

7.3 Le « *PRINT* » en assembleur

Je vous propose une petite routine très simple et fort utile en mode texte. Elle permet d'afficher une chaîne de caractères à l'écran en 32, 40 ou 80 colonnes :

```
ECRIRE:    LD A, (HL)
           CP 0
           RET Z
           CALL 0A2H
           INC HL
           JR ECRIRE
```

Il suffit de charger dans HL l'adresse du premier caractère de la chaîne à afficher. Puis d'appeler la routine, comme dans l'exemple suivant :

```
EXEMPL:    LD HL, DATA
           CALL ECRIRE
           RET
DATA:      DB 'COUCOU', 0
           ;
```

Ne pas oublier de mettre un 0 à la fin de la chaîne. Cette routine comme le PRINT du Basic gère les codes CTRL et ESC.

7.4 Système a disquettes ou pas ?

Pour savoir si l'ordinateur possède un lecteur de disquettes ou non, il faut vérifier l'état d'un Hook modifié par le système d'exploitation de disquette. Le Hook H.PHYD convient parfaitement. Ainsi, si 0FFA7h (adresse de H.PHYD) contient 0C9h (code du RET en Z80), vous travaillez sur un système à cassette. Si 0FFA7h contient autre chose que 0C9h (qui devrait être 0F7h, code du RST 30), alors l'ordinateur possède un lecteur de disquette, intégré ou non.

De plus, lorsqu'un lecteur de disquette est présent, certaines variables système sont modifiées :

adresse	contenu
0FB21h	nombre de lecteurs sur le premier contrôleur
0FB22h	numéro de Slot du premier contrôleur
0FB23h	nombre de lecteurs sur le second contrôleur
0FB24h	numéro de Slot du second contrôleur
0FB25h	nombre de lecteurs sur le troisième contrôleur
0FB26h	numéro de Slot du troisième contrôleur
0FB27h	nombre de lecteurs sur le quatrième contrôleur
0FB28h	numéro de Slot du quatrième contrôleur

Un MSX peut donc gérer jusqu'à 8 lecteurs de disquettes simultanément. S'il y a moins de quatre

contrôleurs, les variables système correspondant aux contrôleurs inexistantes contiennent 0.

En cas d'absence de lecteur de disquette, cette zone mémoire n'est pas initialisée et contient donc n'importe quoi.

7.5 Traiter les erreurs liées aux disquettes

Il est fort désagréable de voir s'afficher, au milieu d'un programme en français, un message du genre « Disk write protected » ou « Disk offline ». Il est heureusement possible de remédier à ce genre de désagrément. Voici la marche à suivre :

1. Modifier l'adresse contenue en 0F323h et 0F324h pour détourner le traitement des erreurs disque vers votre propre routine.
2. Votre routine pourra envoyer les messages adéquats sachant que l'accumulateur contient le numéro du lecteur, alors que le bit 0 du registre C indique si l'erreur s'est produite lors d'une lecture (bit à 0) ou d'une écriture (bit à 1). Les bits 1, 2 et 3 indiquent le type d'erreur selon la table suivante :

3	2	1	Type d'erreur
0	0	0	Protégée contre l'écriture (« Write protected »)
0	0	1	Disque pas prêt (« Disk offline »)
0	1	0	Erreur de CRC (« CRC error »)
0	1	1	Erreur de lecture (« Seek error »)
1	0	0	Fichier non trouvé (« File not found »)
1	0	1	Erreur d'écriture (« Write error »)
1	1	0	Autre erreur

3. Votre routine doit rendre le contrôle au DOS par un RET après avoir chargé le registre C avec l'action à exécuter (0 = Ignorer ; 1 = Recommencer ; 2 = Annuler).

7.6 Faire de la musique en assembleur

La méthode classique pour faire de la musique en assembleur consiste à utiliser une interruption pour charger les registres du PSG 8910 à intervalle régulier.

Le programmeur peut aussi utiliser le système des queues musicales. Malheureusement, cette méthode est difficile à mettre en œuvre.

Il existe une méthode très simple pour exécuter en langage machine l'équivalent de l'instruction PLAY du Basic. Mais cette méthode N'EST PAS GARANTIE officiellement par Microsoft et ASCII. Quelque peu empirique, elle est cependant officieusement garantie, puisqu'ASCII a affirmé qu'elle ne serait pas modifiée. Cela se vérifie pour l'instant puisque tous les MSX vendus en France à ce jour exécutent parfaitement cette routine. Vous utilisez cette routine à vos risques et périls quant à une éventuelle compatibilité avec les MSX à venir.

Pour exécuter cette routine, il faut charger le registre double HL avec l'adresse du premier octet de la

chaîne de caractères à jouer (la chaîne est la même qui pour un PLAY en Basic), puis appeler la routine située à l'adresse 073E5h en Main-ROM. Voici un exemple en assembleur :

```
                ORG 0C000H
;
PLAY            EQU 073E5H
;
                LD HL, DATA
                CALL PLAY
                RET
;
DATA:          DB 022H
                DB 'O5L6DABDACF'
                DB '"', '"'
                DB 'O3L4DEDE'
                DB '"', '"'
                DB 'O7L8ADEFAGED'
                DB 022H, 000H
```

Attention à ne pas oublier le 0 en fin de chaîne. Dans l'exemple ci-dessus, on présuppose que la Main-ROM est sélectionnée.

7.7 *Passage de paramètres du Basic au langage machine*

Dans la plupart des applications, il n'est pas indispensable que l'intégralité du programme soit écrit en langage machine. Il suffit bien souvent de quelques routines en Z80 pour donner un look professionnel à un programme en Basic. Dans ces cas, le problème de l'échange d'informations entre Basic et langage machine se pose.

La méthode classique consiste à définir une zone de communication à laquelle on accède sous Basic par les instructions POKE et PEEK. Cette méthode présente plusieurs inconvénients : l'exécution est plutôt lente, les instructions sont longues et occupent un précieux espace mémoire, toutes les données sont enregistrées deux fois (donc perte très importante de place mémoire).

La fonction USR du Basic (que la majorité des utilisateurs, y compris moi, emploient comme une simple instruction CALL) permet l'échange automatique de données dans les 2 sens. Voyons dans le détail le fonctionnement de la fonction USR. Plusieurs cas se présentent suivant la nature du paramètre à passer :

Le paramètre entre parenthèse est un entier :

C'est le cas le plus simple. La fonction USR met 2 dans la variable système VALTYP (0F663h) pour indiquer qu'il s'agit d'un entier. Le registre A contient aussi cette valeur (2) à l'entrée de votre routine. La valeur du paramètre se trouve sur 2 octets dans la variable système DAC+2 (0F7F6H+2). Le registre HL donne l'adresse de la variable DAC.

Prenons un exemple : Nous désirons réaliser une routine en langage machine qui multipliera automatiquement un entier positif par 2.

```
                ORG 0C000H
;
```

```

DEBUT:      CP 2          ; Avons-nous bien affaire à un entier ?
            RET NZ       ; Retour au Basic si ce n'est pas le cas
;
            INC HL
            INC HL       ; HL = position de l'entier
            XOR A        ; carry à 0
            RL (HL)      ; décalage 8 bits de poids faible(*2)
            INC HL
            RL (HL)      ; décalage 8 bits de poids faible(*2)
            RET NC       ; Retour si il n'y a pas de dépassement
;
ERROR:      LD HL,DATA   ; HL = Adresse message
            CALL ECRIRE  ; à l'écran
            RET
;
NONENT:     LD HL,DATA   ; HL = Adresse message
            CALL ECRIRE  ; à l'écran
            RET
;
DATA:       DB 'Overflow',0
;
;
ECRIRE:     CALL 0A2H
            INC HL
            LD A, (HL)
            CP 0
            RET Z
            JR ECRIRE

```

Une fois la routine assemblée, lancer la par le programme Basic suivant :

```

10 CLEAR,&HC000:BLOAD"Routine.bin"
20 A%=2:DEFUSR=&HC000:PRINT USR(A%)

```

Le paramètre est en simple précision :

C'est un peu plus compliqué. La fonction USR met 4 dans la variable système VALTYP (0F663h) pour indiquer qu'il s'agit d'une valeur en simple précision. Le registre A contient aussi cette valeur (4) à l'entrée de votre routine. La valeur du paramètre se trouve sur 4 octets dans la variable système DAC (0F7F6h). Le registre HL donne l'adresse de la variable DAC.

Format d'une valeur en simple précision :

Octet 1 :

SM	SE	EX5	EX4	EX3	EX2	EX1	EX0
----	----	-----	-----	-----	-----	-----	-----

EX0 à EX5 = Valeur de l'exposant (0 ~ 31).

SE = Signe de l'exposant. 0 pour négatif ; 1 pour positif.

SM = Signe de la mantisse. 0 pour positif ; 1 pour négatif.

Octet 2 :

PC3	PC2	PC1	PC0	SC3	SC2	SC1	SC0
-----	-----	-----	-----	-----	-----	-----	-----

SC0 à SC3 = Second chiffre en BCD.

PC0 à PC3 = Premier chiffre en BCD.

Octet 3 :

TC3	TC2	TC1	TC0	QC3	QC2	QC1	QC0
-----	-----	-----	-----	-----	-----	-----	-----

QC0 à QC3 = Quatrième chiffre en BCD.

TC0 à TC3 = Troisième chiffre en BCD.

Octet 4 :

CC3	CC2	CC1	CC0	SC3	SC2	SC1	SC0
-----	-----	-----	-----	-----	-----	-----	-----

SC0 à SC3 = Sixième chiffre en BCD.

CC0 à CC3 = Cinquième chiffre en BCD.

Par exemple - 3483200 serait codé :

$$-3483200 = -0,348320 * 10^7$$

donc

le premier octet contient 11000111B, soit 0C7h

le second octet code les 2 premiers chiffres, soit 034h

le troisième octet code les 2 chiffres suivants, soit 083h

le quatrième octet code les deux derniers chiffres, soit 020h

Le paramètre est en double précision :

la fonction USR met 8 dans la variable système VALTYP (0F663h) pour indiquer qu'il s'agit d'une valeur en double précision. Le registre A contient aussi cette valeur (8) à l'entrée de votre routine. La valeur du paramètre se trouve sur 8 octets dans la variable système DAC (0F7F6h). Le registre HL donne l'adresse de la variable DAC.

Les 8 octets codent le nombre en double précision de la même manière qu'en simple précision (voir ci-dessus pour plus de précisions), si ce n'est que la mantisse est sur 7 octets au lieu de 3.

Le paramètre est une chaîne de caractères :

La fonction USR met 3 dans la variable système VALTYP (0F663h) pour indiquer qu'il s'agit d'une chaîne de caractères. Le registre A contient aussi cette valeur (3) à l'entrée de votre routine. Le premier octet de la variable système DAC (0F7F6h) donne la longueur de la chaîne. Les deux octets suivants indiquent l'adresse où se trouve la chaîne. Le registre DE donne l'adresse de la variable DAC.

Voici un exemple qui transforme tous les caractères majuscules en minuscules dans une chaîne :

```
ORG 0C000H
;
CP 3
RET NZ ; Retour au Basic si ce n'est pas une chaîne
;
LD A, (DE) ; Longueur de la chaîne
OR A
```

```

RET Z                ; Retour au Basic si longueur = 0

LD B,A              ; Nombre de caractères dans B pour DJNZ
CONVC: INC DE
LD A,(DE)           ; Prend un caractère dans la chaîne
CP 041H
JR C,NOCHAR        ; Si A < 041H, ce n'est pas une majuscule
CP 05AH
JR NC,NOCHAR       ; Si A > 064H, ce n'est pas une majuscule
OR 020H            ; Convertie la majuscule en minuscule
LD (DE),A          ; Place le caractère dans la chaîne
NOCHAR: DJNZ CONVC
RET                ; retour au Basic

```

Après avoir tapé et assemblé cette routine au format binaire, entrez le programme Basic suivant et faites RUN :

```

10 CLS
20 CLEAR 200,&HC000
30 BLOAD".bin":DEFUSR=&HC000
40 A$="TRANSFORMATION":A$=USR(A$)
50 PRINT A$

```

7.8 Ajouter une instruction au Basic avec CMD ou IPL

A partir de la version 2.0, le Basic possède deux instructions CMD et IPL qui ne font rien excepté un appel à leur propre Hook. IPL appelle H.IPL (0FE03h) et CMD appelle H.CMD (0FE0Dh) afin que l'on puisse créer nos propres instructions.

Lors de l'appel au Hook, le registre HL du CPU contient le pointeur de l'interpréteur Basic dans le buffer de l'instruction en cours d'exécution. La dernière valeur mise dans la pile contient les valeurs des registres AF (F contenant les indicateurs d'erreurs de l'interpréteur).

Il est possible d'ajouter des paramètres derrière ces deux instructions mais cela nécessite une connaissance avancée de l'interpréteur du Basic car le pointeur ne pointe pas un simple texte ASCII mais un texte codé par l'interpréteur du Basic. Par exemple, si vous entrez « CMD PRINT », le registre HL pointera le code de l'instruction PRINT (091h) et non pas le texte 070h, 072h, 069h 06Eh, 074h (« PRINT »).

Pour créer votre instruction, vous disposez des trois routines suivantes en ROM :

04C64h (Main-ROM) **FRMEVL**

Rôle : Envoie la valeur au pointeur dans la variable système DAC au format approprié.
 Entrée : HL = Pointeur actuel.
 Sortie : HL = Pointeur placé sur l'octet suivant la valeur.
 VALTYP (0F663h) = 2, 3, 4 ou 8 selon le type de la valeur.
 DAC (0F7F6h) = Valeur convertie au format approprié.
 Modifie : AF, HL.

0542Fh (Main-ROM) *FRMQNT*

Rôle : Convertie la valeur au pointeur au format simple précision (sur 2 octets).
Entrée : HL = Pointeur actuel.
Sortie : HL = Pointeur placé sur l'octet suivant la valeur.
DE = Valeur codée sur 2 octets.
Modifie : AF, HL et DE.
Note: Si la valeur dépasse, il y aura retour au Basic avec l'erreur "Overflow".

Voici un exemple avec CMD :

< Donner la fonction CAPS ON/OFF à l'instruction CMD du Basic. >

```
;
; Instruction CMD CAPS routine
;
CHGCAP EQU 0132H ; LED CAPS ON/OFF
CAPST EQU 0FCABH ; Statut de CAPS
HCMD EQU 0FE0DH ; CMD Hook

ORG 0D000H-7 ; Adresse de la routine - Taille Header
;
; Header
;
db 0feh
dw Debut
dw Fin
dw Debut
;
; Détournement du Hook CMD
;
Debut: LD BC,5 ; place les données du nouveau Hook
LD DE,HCMD
LD HL,HDAT
LDIR
RET
;
; Donnée de la routine de détournement (5 octets)
;
HDAT: JP CAPKEY
NOP
NOP
;
; Routine exécutée par l'instruction CMD
;
CAPKEY:
CP 043H ; Teste si le premier caractère est "C"
RET NZ
RST 10 ; INC HL possible
LD A,(HL)
CP 041H ; Teste si le second caractère est "A"
RET NZ
RST 10 ; INC HL possible
LD A,(HL)
```

```

        CP    050H        ; Teste si le second caractère est "P"
        RET    NZ
        RST    10        ; INC HL possible
        LD    A, (HL)
        CP    053H        ; Teste si le second caractère est "S"
        RET    NZ

        LD    A, (CAPST)
        CPL
        LD    (CAPST), A
        AND    1
        CALL  CHGCAP
RET BASIC:
        POP    AF        ; Pour ne pas avoir d'erreur
        RST    10        ; Pointeur à la fin de l'instruction
        RET

fin:

```

Une fois la routine assemblée et sauvegardé au format binaire sous le nom "CMDCAPS.BIN", entrer la ligne suivante pour activer l'instruction.

```
CLEAR300, &HD000:BLOAD"CMDCAPS.BIN", R
```

Ensuite, vous pourrez entrer l'instruction CMD CAPS pour allumer ou éteindre la LED de la touche CAPS. Cette exemple marche parce que le Mot « CAPS » ne contient aucun mot clé du Basic.

Autre exemple :

< Créer une instruction pour changer le CPU du MSX turbo R. >

```

;
; Utilisation : CMD Z80 ou CMD R800 sous MSX-Basic
;
CHG CPU EQU 0180h        ; Change le CPU Z80/R800
HCMD EQU 0fe0dh         ; CMD Hook

                ORG 0d000h-7 ; Adresse de la routine - 7
;
; Header (taille = 7 octets)
;
        DB    0feh
        DW    Debut
        DW    Fin
        DW    Debut
;
; Détournement de l'instruction CMD via le Hook
;
Debut:        LD    BC, 5
                LD    DE, HCMD
                LD    HL, HDAT
                LDIR          ; place les données du nouveau Hook
                RET
;
; Nouvelles données du HOOK (5 octets)
;
HDAT:         JP    R800ROM
                NOP
                NOP

```

```

;
; CMD Z80 ou CMD R800? (Routine exécutée par CMD)
;
R800ROM:  CP    5AH          ; Teste si le premier caractère
          JR    Z,Z80MODE  ; du paramètre est "Z"
          CP    052H       ; Teste si c'est "R"
          RET   NZ
          RST  10          ; INC HL possible
          LD   A,(HL)
          CP    038H       ; Teste si le second caractère est "8"
          RET   NZ
          RST  10          ; INC HL possible
          LD   A,(HL)
          CP    030H       ; Teste si le troisième est "0"
          RST  10
          RST  10          ; INC HL possible
          LD   A,(HL)
          CP    030H       ; Teste si le quatrième est "0"
          RET   NZ
          LD   A,081H      ; R800 mode ROM
          JR   RETBASIC
Z80MODE:  RST  10          ; INC HL possible
          LD   A,(HL)
          CP    038H       ; Teste si le second caractère est "8"
          RET   NZ
          RST  10          ; INC HL possible
          LD   A,(HL)
          CP    030H       ; Teste si le troisième est "0"
          RET   NZ
          LD   A,080H      ; Z80
RETBASIC: CALL  CHGCPU     ; Change le CPU
          POP  AF          ; Pour ne pas avoir d'erreur
          RST  10          ; Pointeur à la fin de l'instruction
          RET
Fin:

```

Une fois la routine assemblée et sauvegardée au format binaire sous le nom "CMDR800.bin", entrer la ligne suivante pour initialiser la nouvelle instruction.

```
CLEAR300,&HD000:BLOAD"CMDR800.bin",R
```

Ensuite, vous pourrez entrer l'instruction CMD R800 pour activer le mode R800 ROM ou CMD Z80 pour activer le mode Z80.

7.9 Les codes de contrôle CTRL

Chacun sait que sous Basic, en enfonçant simultanément les touches « CTRL » et « L », on efface l'écran. Le MSX dispose de plusieurs autres codes du même type qui peuvent être utilisés en mode direct ou dans un programme. En voici la liste :

code	touches	effet
2	CTRL+B	place le curseur au début du mot précédent
3	CTRL+C	interrompt le programme (BREAK)
5	CTRL+E	efface toute la ligne à droite du curseur
6	CTRL+F	place le curseur au début du mot suivant
7	CTRL+G	émission d'un bref bip sonore
8	CTRL+H	back space (BS)
9	CTRL+I	tabulation / idem touche TAB
10	CTRL+J	descend le curseur d'une ligne
11	CTRL+K	idem touche EFE ou HOME
12	CTRL+L	effacement de l'écran
13	CTRL+M	effectue une entrée / idem touche RETURN
14	CTRL+N	place le curseur en fin de ligne
18	CTRL+R	idem touche INS
21	CTRL+U	efface la ligne où se trouve le curseur
24	CTRL+X	idem touche SELECT
27	CTRL+[idem touche ESC
28	CTRL+\	idem flèche droite
29	CTRL+]	idem flèche gauche
30	CTRL+^	idem flèche haut
31	CTRL+_	idem flèche bas

Essayez donc cet exemple (MSX2) :

```
10 SCREEN 0: WIDTH 80: COLOR 10,0,0: CLS
20 A$="Ce texte s'affiche bizarrement !!!"
25 LOCATE 25,10
30 FOR I=1 TO LEN (A$)
40 PRINT CHR$(30)+MID$(A$, I, 1) ;
50 NEXT I
```

7.10 Les codes escape [ESC]

Le MSX peut utiliser toute la série des codes ESC compatible avec les terminaux VT-52 ou HEATH-19 dont la liste suit :

code	effet
ESC A - 27 65	monte le curseur d'une ligne
ESC B - 27 66	descend le curseur d'une ligne
ESC C - 27 67	déplace le curseur vers la droite
ESC D - 27 68	déplace le curseur vers la gauche
ESC H - 27 72	curseur en haut à droite (HOME)
ESC Y col ligne	curseur en X, Y (LOCATE) voir ci-dessous
ESC j - 27 106	efface l'écran (CLS)
ESC E - 27 69	efface l'écran (CLS)
ESC K - 27 75	efface jusqu'à la fin de la ligne
ESC J - 27 74	efface jusqu'à la fin de l'écran
ESC I - 27 108	efface toute la ligne
ESC L - 27 76	insère une ligne
ESC M - 27 77	détruit une ligne
ESC x4 - 27 120 52	définit un curseur de type entier
ESC x5 - 27 120 53	éteint le curseur
ESC y4 - 27 121 52	définit un curseur de type barre
ESC y5 - 27 121 53	allume le curseur

La séquence ESC Y est légèrement plus compliquée à utiliser. Il faut en effet envoyer les deux octets de ESC Y (27 et 89), puis les faire suivre par deux octets qui définissent la colonne (n° de colonne + 020h) ainsi que la ligne (n° de la ligne + 020h). Affichons par exemple la chaîne « ICI » à la position (10, 33) :

```
10 PRINT CHR$(27)+"Y*AIICI"
```

On envoie un CHR\$(27) puis « Y » suivi de « * » (code ASCII = 42, soit 10 + 020h), et « A » code ASCII = 65 soit 33 + 020h).

Très peu de gens connaissent l'existence de ces codes ESC qui peuvent pourtant parfois rendre de grands services comme le démontre l'exemple suivant :

```
10 SCREEN 0: WIDTH 40: COLOR 10,0,0: CLS: KEYOFF
20 E$=CHR$(27)
30 FOR I=0 TO 17: PRINT"LIGNE";I: NEXT
40 PRINT: PRINT"On peut effacer les dernières lignes.": PRINT
50 PRINT"Comme dans une fenêtre...": PRINT
60 PRINT TAB(5);"... aussi rapidement qu'avec un CLS";
70 FOR I=1 TO 1500: NEXT
80 LOCATE 0,19: PRINT"Sans toucher aux autres !"+E$+"J"
90 FOR I=1 TO 1500: NEXT I: LOCATE 0,18: GOTO 40
```

Essayez d'enlever +E\$+"J" à la ligne 80.

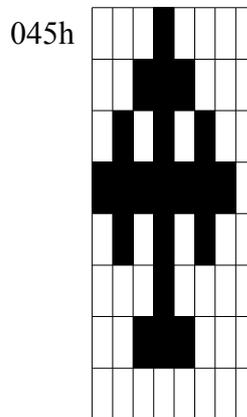
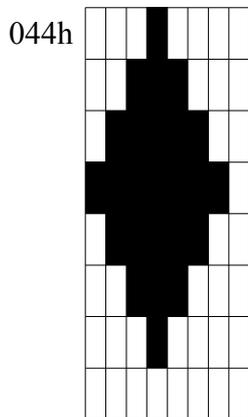
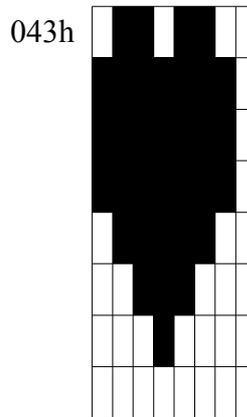
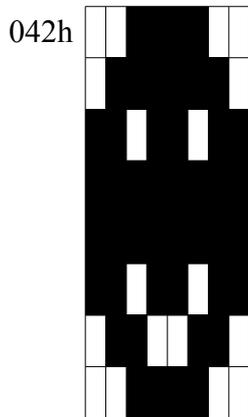
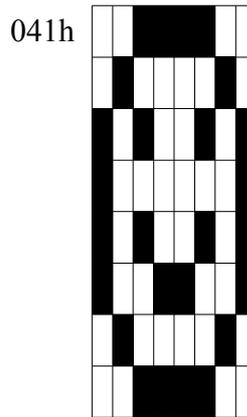
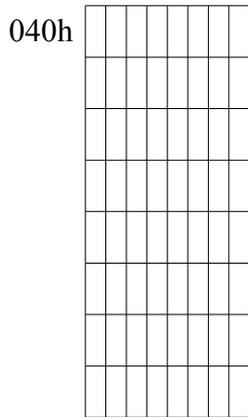
7.11 Utiliser le second jeu de caractères

Les MSX possèdent un second jeu de caractères. Ce dernier est réduit (32 caractères) et on y accède par un CHR\$(1) suivi d'un code ASCII entre 64 (040h) et 95 (05Fh). Par exemple, en SCREEN 1, la ligne :

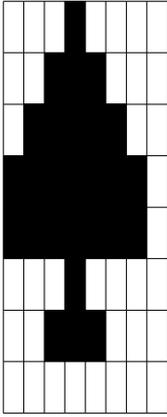
```
PRINT CHR$(1)+"A"
```

fera s'afficher à l'écran une petite tête sympathique. Notez que tous ces caractères se trouvent définis dans des matrices 8 sur 8. On perdra donc, en SCREEN 0, les 2 bits de poids faible, et le caractère sera amputé de ses 2 colonnes les plus à droite.

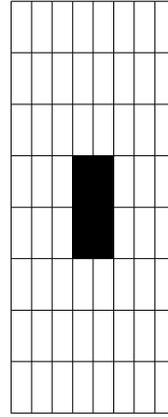
Voici les matrices des caractères de ce second jeu :



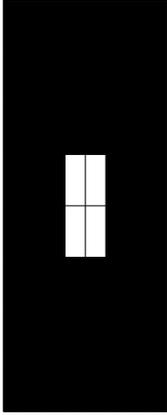
046h



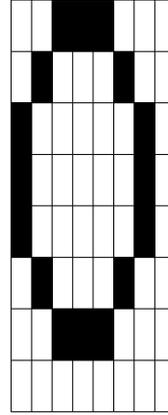
047h



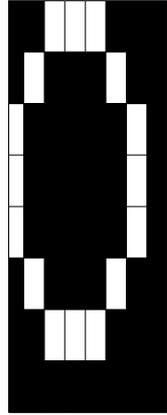
048h



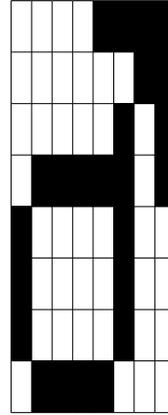
049h



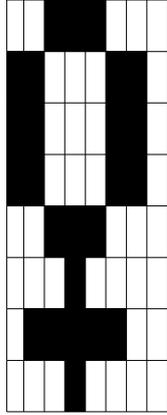
04Ah



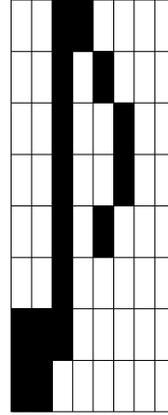
04Bh



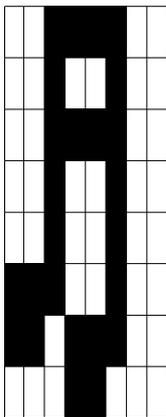
04Ch



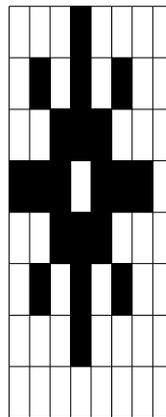
04Dh



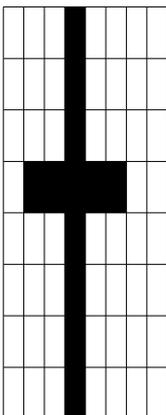
04Eh



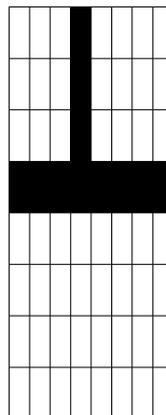
04Fh



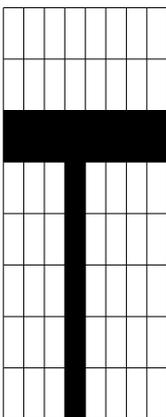
050h



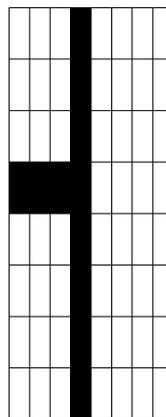
051h



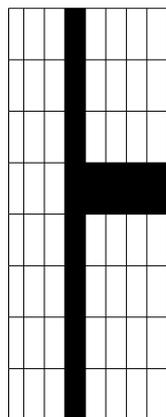
052h



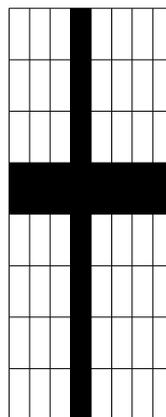
053h



054h



055h




```

                LD B, (HL)
                CP B
                POP BC
                JR Z, RAM
                DJNZ LOOP
;
RIEN:          NOP
                JR RETOUR
;
RAM:          NOP
                JR RETOUR
;
MNROM EQU     0FCC1H
RETOUR:       LD A, (MNROM)
                LD HL, 04000H
                CALL ENASLT
                RET

```

Le programme procède de la manière suivante : il sélectionne tous les Slots - primaires et secondaires - un par un. Il écrit une donnée puis la relit. Si la donnée lue est identique à celle écrite, alors le Slot actuel contient de la mémoire vive, le programme s'arrête. Dans le cas contraire, on passe au Slot suivant.

A vous de compléter les routines « RIEN : » et « RAM : » en fonction de votre application.

L'utilisation :

Sur MSX1, pas de problème, vous avez trouvé de la mémoire vive, utilisez la. Par contre à partir du MSX2, il faut faire attention à plusieurs choses :

- Un disque virtuel peut être implanté dans la mémoire vive que vous venez de trouver. Si tel est le cas, la variable système SLTWRK (0FD09h) contient le numéro du Slot dans lequel se trouve le disque virtuel sous la forme FXXXSSPP en binaire.

F indique le type de Slot. (0 pour Slot Primaire ; 1 pour Slot Secondaire.)

SS indique le numéro de Slot Secondaire (0 ~ 3).

PP indique le numéro de Slot Primaire (0 ~ 3).

Nous verrons la signification des bits X plus bas, masquez ces bits pour pouvoir comparer avec le numéro de Slot retourné par la routine qui est donnée plus haut. Cependant, il faut faire attention car si SLTWRK donne 0XXX0011 (Slot primaire 3), ma routine renverra 1XXX1111 (Slot secondaire 3, primaire 3). Le résultat est équivalent lors d'un appel inter-Slot ou d'un ENASLT. Pour simplifier la programmation, ma routine donne TOUJOURS le résultat sous forme de Slot secondaire même s'il s'agit d'un Slot primaire. Il va de soi que s'il n'y a pas de disque virtuel, ces 5 bits seront tous à 0.

Le bit 6 de SLTWRK est un indicateur binaire. S'il est à 0, le disque virtuel n'est pas initialisé; au contraire, le disque virtuel est en place si ce bit est à 1.

Si le disque virtuel est installé, vous pouvez encore disposer de la mémoire vive non utilisée par celui-ci. Il suffit de savoir où s'arrête le disque virtuel. Les cases mémoires 00000H et 00001H contiennent l'adresse de début de la zone mémoire laissée libre par le disque virtuel. Les adresses 00002H à 0007FH ne sont pas utilisées sur MSX2, elles sont réservées aux futures versions du système MSX. Ne pas y toucher. Le disque virtuel commence en 00080h.

Un autre logiciel utilise peut-être déjà la mémoire que vous avez trouvée. Si tel est le cas, le bit 5 de SLTWRK sera à 1. Si le bit est à 0, vous même - si vous décidez d'utiliser la mémoire vive trouvée pour votre application - devez mettre le bit 5 de SLTWRK à 1.

Voici un résumé des fonctions de SLTWRK sur MSX2 :

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
F	RMD	APP	RES	SS1	SS0	PP1	PP0

PP0 et PP1 = Numéro de Slot primaire.

SS0 et SS1 = Numéro de Slot secondaire.

RES = Bit réservé.

APP = Indicateur d'application. 1 = RAM déjà utilisée ; 0 = RAM disponible.

RMD = Indicateur de RAM-Disk. 1 = Utilisée par le RAM-Disk ; 0 = RAM disponible.

F = Type de Slot ; 0 = Primaire ; 1 = Secondaire.

7.13 *Détourner le Reset*

Presque tous les MSX vendus en France possèdent un bouton RESET (le V20 de Canon et le YC-64 Yashica sont les seuls MSX sans RESET ayant connu une grande diffusion). Le programmeur peut interdire l'accès à son application à l'utilisateur final en détournant le RESET.

Le principe est assez simple. On installe en mémoire vive des codes qui font croire au MSX qu'il ne s'agit pas de RAM mais d'une cartouche. En effet, dans ce dernier cas, le MSX passe toujours la main à la cartouche (sinon le programme en cartouche ne démarrerait pas automatiquement). Etant trompé (comme nous sommes diaboliques), le MSX va donc rendre la main au programme que nous aurons installé auparavant.

On utilise le programme du paragraphe précédent pour trouver la RAM en page 1 (04000h à 07FFFh). Le reste du programme est simple à comprendre :

```

                                ORG 0C000H
;
ENASLT    EQU 00024H
FIND:     LD B,0FH
          LD HL,04000H          ; Plage 1
LOOP:     LD A,B
          OR 080H
          PUSH BC
          PUSH AF
          PUSH HL
          CALL ENASLT          ; Sélection de Slot
          POP HL
          POP AF
          LD (HL),A
          LD B,(HL)
          CP B
          POP BC
          JR Z,RAM
          DJNZ LOOP           ; Cherche la RAM
;
RIEN:     JR RETOUR
;
RAM:      LD IX,04000H
```

```

                LD (IX+0),041H           ; Code "A"
                LD (IX+1),042H           ; Code "B"
                LD (IX+2),000H           ; Adresse de
                LD (IX+3),0C1H           ; départ
                JR RETOUR

;
MNROM           EQU 0FCC1H
RETOUR:        LD A, (MNROM)
                LD HL,04000H             ; Replace la Main-ROM
                CALL ENASLT              ; en page 1.
                RET

;
;
;
                ORG 0C100H

;
BREAKX         EQU 000B7H
INITXT         EQU 0006CH

;
                CALL INITXT              ; SCREEN 0, 40 colonne
                LD HL,MESS
                CALL ECRIRE
WAIT:          CALL BREAKX              ; Teste CTRL+STOP
                RET C
                JR WAIT
ECRIRE :       LD A, (HL)
                CP 0
                RET Z
                CALL 0A2H
                INC HL
                JR ECRIRE
MESS:          DB 01BH, 'Y*&'
                DB 'LE RESET NE REND'
                DB ' PAS LA MAIN !',0

```

Dans l'exemple ci-dessus, tapez et assemblez le programme, puis lancez l'exécution en 0C000h. A partir de ce moment, chaque fois que vous appuierez sur le bouton RESET, le programme affichera le message « Le reset ne rend pas la main ! ». Il suffit d'enfoncer les touches CTRL et STOP pour récupérer le contrôle.

Notez que l'on utilise un code ESC pour positionner le curseur (première ligne de « MESS: »).

Une cartouche se distingue des autres supports. En effet, toute cartouche contient « AB » (codes 041H et 042h) comme deux premiers octets (en général c'est aux adresses 04000H et 04001h). Elle renferme ensuite (en 04002H et 04003h) l'adresse de démarrage du programme en cartouche.

7.14 Ajouter des mots clés au Basic

Tant qu'on y est, continuons à simuler le fonctionnement d'une cartouche. Vous savez sans doute qu'une cartouche peut parfois contenir des extensions au MSX-Basic. On accède à ces nouvelles instructions par l'instruction CALL (ou « _ ») suivi d'un nom et éventuellement de paramètres.

L'adresse de la routine de traitement des mots clefs est donnée par le 5ème et le 6ème octet (adresse 04004H et 04005h) de la cartouche. A chaque fois que le Basic trouve un CALL, il appelle la routine à cette adresse. Le mot clef se trouve alors sur 16 octets dans la zone des variables système (variable

PROCNM, adresses 0FD89h à 0FD98h). Le registre double HL pointe sur le premier caractère non blanc (code 020h) après le mot clef, ce qui permet de récupérer les paramètres. Il faut réactualiser HL de manière à poursuivre l'exécution du programme Basic après traitement du CALL. Il suffit pour cela d'incrémenter HL jusqu'au moment où HL pointe sur un 0 (code de fin de ligne Basic) ou 03Ah (code des deux points « : », séparant deux instructions Basic). De plus, votre routine doit toujours rendre la main avec l'indicateur Carry à 0. Dans la routine de traitement, tous les registres peuvent être utilisés (sauf SP, bien sûr). Si le mot clef est inconnu, il faut mettre l'indicateur Carry à 1, puis rendre la main à l'interpréteur par un RET, sans avoir modifié HL.

Dans l'exemple suivant, nous créons un mot clef « FILLSCREEN » qui remplit l'écran (en SCREEN 0 uniquement, 40 ou 80 colonnes) avec le caractère qui suit :

```
10 _FILLSCREEN("Z"):BEEP
```

Cet exemple fonctionne parfaitement sur tous MSX.

```

                ORG 0C000H

;
ENASLT         EQU 00024H
FIND:          LD B,0FH
                LD HL,04000H
LOOP:          LD A,B
                OR 080H
                PUSH BC
                PUSH AF
                PUSH HL
                CALL ENASLT
                POP HL
                POP AF
                LD (HL),A
                LD B,(HL)
                CP B
                POP BC
                JR Z,RAM
                DJNZ LOOP

;
RIEN:          JR RETOUR

;
RAM:           LD IX,04000H
                LD (IX+0),041H
                LD (IX+1),042H
                LD (IX+2),000H
                LD (IX+3),000H
                LD (IX+4),000H
                LD (IX+5),0C1H
                JR 00000H

;
MNROM          EQU 0FCC1H
RETOUR:        LD A,(MNROM)
                LD HL,04000H
                CALL ENASLT
                RET

;
;
;
                ORG 0C100H

;
FILVRM         EQU 00056H
                PUSH HL

```

```

                LD HL,WORD
                LD DE,0FD89H
LOOP2:         LD A,(DE)
                LD B,(HL)
                CP B
                JR NZ,SYNERR
                CP 0
                INC HL
                INC DE
                JR Z,OUT
                JR LOOP2

;
SYNERR:       POP HL
                XOR A
                CCF
                RET

;
OUT:          POP HL
                PUSH HL
                LD A,(HL)
                CP 028H
                JR NZ,SYNERR
                INC HL
                LD A,(HL)
                CP 022H
                JR NZ,SYNERR
                INC HL
                LD B,(HL)
                INC HL
                LD A,(HL)
                CP 022H
                JR NZ,SYNERR
                INC HL
                LD A,(HL)
                CP 029H
                JR NZ,SYNERR
LOOP3:       INC HL
                LD A,(HL)
                CP 020H
                JR Z,LOOP3
                CP 0
                JR Z,SUITE
                CP 03AH
                JR NZ, SYNERR

;
SUITE:       PUSH HL
                LD HL,0
                LD A,B
                LD BC,00800H
                CALL FILVRM
                POP HL
                POP BC
                XOR A
                RET

;
WORD:        DB 'FILLSCREEN',000H

```

Après avoir tapé et assemblé le programme ci-dessus, lancez l'exécution en 0C000h. Le MSX fera

automatiquement un RESET (pour que la recherche de la cartouche se produise). Lors du retour sur Basic, tapez CLEAR 200, &HC100

A partir de là, vous pouvez à tout moment appeler la nouvelle fonction par un CALL FILLSCREEN (“caractère”).

Pour créer une autre fonction, il suffit de modifier le programme à partir du label « SUITE : »

7.15 Manipuler la souris

Je vous propose un sous-programme en assembleur qui permet d'utiliser la souris depuis le Basic.

```
                ORG 0C000H
;
EXTROM          EQU 0015FH
NEWPAD          EQU 001ADH
GTTRIG          EQU 000D8H
NWRVRM          EQU 00177H
;
SOURIS:         LD A,0CH
                LD IX,NEWPAD
                CALL EXTROM
                LD A,0DH
                LD IX,NEWPAD
                CALL EXTROM
                LD B,A
                LD A,(X)
                ADD A,B
                LD (X),A
;
                LD A,0CH
                LD IX,NEWPAD
                CALL EXTROM
                LD A,0EH
                LD IX,NEWPAD
                CALL EXTROM
                LD B,A
                LD A,(Y)
                ADD A,B
                LD (Y),A
;
                CALL Sprite
                LD A,1
                CALL GTTRIG
                CP 0FFH
                RET Z
                JR SOURIS
;
Sprite:         LD A,(X)
                LD HL,0FA01H
                CALL NWRVRM
                RET
;
X:              DB 000H
Y:              DB 000H
```

Voici un exemple d'exploitation de la routine ci-dessus en Basic :

```

10 SCREEN 7: SET PAGE 0,0: COLOR 10,0,0: CLS: X=&HC045: Y=X+1
20 Sprite$(0)=CHR$(&HF8)+CHR$(&HC0)+CHR$(&HA0)+CHR$(&H90)+CHR$
(&H88)+CHR$(4)+CHR$(2)
30 PUT Sprite 0,(100,100),7: DEFUSR=&HC000: I=USR(0)
110 PSET(PEEK(&HC054)*2,PEEK(&HC055)+1),14
120 I=USR(0):GOTO 110

```

7.16 Développer un programme en cartouche

Lorsqu'on allume un MSX, le système cherche, avant toutes choses, la RAM dans les Slots puis initialise les variables du système et du Basic. Ensuite, il cherche dans chaque Slot de 0-2 à 3-3 l'entête d'une ROM. Lorsqu'il en trouve une, il lance le programme « Init » de la ROM à l'adresse indiquée dans l'entête.

Une entête se compose de 16 octets placés à 0000h, 4000h, 8000h ou 0C000h de la façon suivante.

Entête	Nom	Description
+0	ID	Les deux premiers octets doivent être 041H et 042H (« AB ») pour indiquer qu'il s'agit d'un ROM additionnelle.
+2	Init	Ces deux octets contiennent l'adresse d'exécution du programme en ROM. Le système sélectionnera la ROM dans le Slot correspondant à l'adresse indiquée et exécutera le programme en ROM. Indiquer 0000h pour ignorer l'exécution.
+4	Statement	Ces deux octets contiennent l'adresse d'exécution d'un programme dont le but est d'ajouter des instructions au Basic en ROM. Ce programme doit résider sur la plage 4000h ~ 7FFFh. Indiquer 0000h si la ROM ne contient pas d'instruction Basic à ajouter.
+6	Device	Adresse d'exécution d'un programme servant à installer un matériel intégré à la cartouche. Indiquer une adresse correspondante à la plage mémoire du Slot. Indiquer 0000h si la cartouche ne contient pas de matériel à installer.
+8	Text	Pointeur du programme Basic contenu en ROM. Indiquer une adresse supérieure à 08000h ou 0C000h selon le Slot utilisé. Indiquer 0000h si la ROM ne contient pas de programme Basic.
+10 ~ 15	Reserved	Réservé. Ces octets devraient normalement être à zéro.

Init

Le programme à Init sera exécuté avant l'installation des périphériques (Disques, RS-232C, etc).

Au moment de l'exécution du programme, seuls les 16Ko du Slot correspondant à l'adresse indiquée dans l'entête est sélectionné. Le registre C contient ce numéro de Slot sous la forme F000SSPP.

Donc si votre ROM a une taille de 32Ko et une entête à 4000h, il faudra sélectionner la deuxième partie de la façon suivante.

```

ld    a,c
ld    h,080h    ; Le registre L n'est pas pris en compte par la
                ; routine de sélection de Slot.

```

```
call ENASLT      ; Sélection du Slot
```

Cette même ROM (de 4000h à 0BFFFh) peut avoir une entête à 8000h. Dans ce cas, la sélection de la partie basse se fait de la façon suivante.

```
ld    a,c
ld    h,040h     ; Le registre L n'est pas pris en compte par la
                ; routine de sélection de Slot.
call  ENASLT     ; Sélection du Slot
```

Statement

En cours...

Device

En cours...

Text

En cours...

8 Le MSX-DOS

Le MSX-DOS nécessite un minimum de 64Ko de RAM. Dans l'environnement du MSX-DOS, la mémoire est répartie de la façon suivante.

MSX-DOS	00000h
Zone libre	00100h
Buffer des FAT	Adresse indiquée à 00006h
Buffer des fichiers	
E/S du secteur	
Zone de buffer de secteur	
Zone de travail des contrôleurs de disques	
Zone de communication fixe du MSX-DOS et du Disk-BASIC	0F1C9h
Variables et zone de travail du système	0F380h
	0FFFFh

La zone de travail des contrôleurs de disques est composée d'une zone de variables de quelques octets suivi du DPB (Disk Parameter Block) de chacun des disques du contrôleur et ce pour chaque contrôleur. Un DPB est une zone de 21 octets comprenant les paramètres d'un disque.

La zone de buffer de secteur prendra la taille du plus grand secteur présent sur un des disques installés.

8.1 Zone fixe de travail et des variables du MSX-DOS1 et du Disk-BASIC

Cette zone contient aussi des Hooks spécifiques au MSX-DOS.

Adresse	Nom	Long.	Fonction
0006h		2	Adresse de la fin de la zone réservée à l'utilisateur. Lorsqu'une commande du MSX-DOS (fichier COM) est lancée, elle se charge à l'adresse 0100h et sa taille peut aller jusqu'à l'adresse indiquée ici.
005Ch	FCBA		Cette zone contient le premier nom de fichier indiqué derrière la commande entrée. Le premier octet indique le numéro de lecteur.
006Ch	FCB2		Cette zone contient le deuxième nom de fichier indiqué derrière la commande entrée. Le premier octet indique le numéro de lecteur.
0080h	DMA		Zone DMA par défaut. Cette zone contient tous paramètres indiqués derrière la commande entrée. Le premier octet indique le nombre de caractères utilisés pour paramètres.
0F1C9h		24	Routine pour afficher une chaîne de caractères précédée du caractère « \$ ». Entrée : DE = Adresse du début de la chaîne.
0F1E2h		6	Routine pour interrompre un programme en cas d'erreur.
0F1E8h		12	Routine d'appel à une routine en Main-RAM sur la page 1 lorsque la Disk-ROM est sélectionnée. Entrée : Contenu de d'adresse pointée par HL = Adresse de la routine à appeler.
0F1F4h		3	Saut à la Routine de changement de nom de fichier. Entrée : HL = Adresse du premier caractère du nom de fichier.
0F1F7h		4	Nom de périphérique « PRN »
0F1FBh		4	Nom de périphérique « LST »
0F1FFh		4	Nom de périphérique « NUL »
0F203h		4	Nom de périphérique « AUX »
0F207h		4	Nom de périphérique « CON ».
0F20Bh		8	Nom trouvé au périphérique. (PRN, LST, NUL, etc)
0F213h		3	Nom d'extension trouvé au périphérique. (PRN, LST, NUL, etc)
0F216h		1	Périphérique actuelle. PRN = -5, LST = -4, ..., CON = -1.
0F221h		2	Date du fichier trouvé.

0F223h		2	Heure du fichier trouvé.
0F22Bh		12	Le tableau contenant le nombre de jours de chaque mois de l'année.
0F237h	BUFINP	4	Zone de travail le la fonction 10 du BDOS.
0F23Bh		1	Indicateur pour envoyer les caractères vers l'imprimante. 0 = Ne pas envoyer ; Autre valeur = Caractères vers l'imprimante.
0F23Dh		2	Adresse actuelle de la DTA.
0F23Fh		4	Numéro du secteur actuel du disque.
0F243h		2	Pointeur à l'adresse du DPB du lecteur actuel.
0F245h		1	Numéro relatif du secteur dans le répertoire. Utilisé par les routines SEARCH FIRST et NEXT. (Ajouter le numéro du premier secteur du répertoire pour connaître le numéro de secteur réel.)
0F246h		1	Numéro du lecteur dont le répertoire a été lu. (0 = « A: », 1 = « B: », etc)
0F247h		1	Numéro du lecteur par défaut. (0 = « A: », 1 = « B: », etc)
0F248h		1	Jour. (Valeur stockée par la commande DATE ou TIME du MSX-DOS.)
0F249h		1	Mois. (Valeur stockée par la commande DATE ou TIME du MSX-DOS.)
0F24Ah		2	Année. 0 = 1980, 1 = 1981, 2 = 1982, etc. (Valeur stockée par la commande DATE ou TIME du MSX-DOS.)
0F24Ch		2	Heure. (?)
0F24Eh		1	Jour de la semaine. 0 = Dimanche, 1 = Lundi, 2 = Mardi, etc. (Valeur stockée par la commande DATE ou TIME du MSX-DOS.)
0F24Fh	H.PROMPT	3	Hook appelé avant l'affichage du message « Insert disk for drive » de l'émulation d'un deuxième lecteur. Lors de l'appel à ce Hook, le registre A contient la lettre du nom de lecteur en code ASCII.
0F252h	H.BDOS	3	Hook appelé avant l'exécution d'une routine BDOS. (évaluation de la fonction appelée.)
0F255h		3	Hook appelé au début de la routine Fix Filename.
0F258h	H.SEARCHDIR	3	Hook appelé au début de la routine SearchDir. (Utile pour changer 0F2DC h afin de trouver aussi les fichiers cachés.)
0F25Bh	H.	3	
0F25Eh	H.	3	Hook appelé au début de la routine NextDirSect.
0F261h	H.	3	Hook appelé au début de la routine de gestion du numéro de

lecteur.

0F264h	H.OPEN	3	Hook appelé au début de la routine Open.
0F267h	H.	3	
0F26Ah	H.GETDPB	3	Hook appelé au début de la routine GetDPB.
0F26Dh	H.CLOSE	3	Hook appelé au début de la routine Close.
0F270h	H.ABSRD	3	Hook appelé au début de la routine RDABS de lecture de secteur(s).
0F273h	H.	3	Hook appelé au début de la routine d'erreur l'accès au disque.
0F276h	H.	3	
0F279h	H.WRABS	3	Hook appelé au début de la routine WRABS d'écriture de secteur(s).
0F27Ch	H.	3	Hook appelé au début de la routine de multiplication. (HL et BC = DE*BC)
0F27Fh	H.	3	Hook appelé au début de la routine de division. (BC = BC/DE, HL = Reste)
0F282h	H.	3	
0F285h	H.	3	
0F288h	H.	3	
0F28Bh	H.	3	
0F28Eh	H.	3	
0F291h	H.	3	
0F294h	H.	3	
0F297h	H.	3	
0F29Ah	H.	3	
0F29Dh	H.	3	
0F2A0h	H.	3	
0F2A3h	H.	3	
0F2A6h	H.	3	
0F2A9h	H.	3	
0F2ACh	H.BUFINP	3	Hook appelé au début de la routine BUFINP.
0F2AFh	H.CONTOUT	3	Hook appelé au début de la routine CONTOUT qui vérifie les codes CTRL.
0F2B2h	H.	3	
0F2B5h	H.	3	Hook appelé au début de la routine de gestion des années (bissextile ou pas).
0F2B8h		1	
0F2B9h		8	Nom du fichier ouvert.
0F2C1h		3	Nom d'extension du fichier ouvert.
0F2C4h		1	Attributs du fichier.

0F2DEh		1	?
0F2DFh		1	?
0F2E1h	DRIVE	1	Lecteur en cours (0 ~ 7). 0 = « A: », 1 = « B: », etc.
0F2E2h		2	Adresse de fin du DMA. (0080h par défaut)
0F2E4h		4	
0F2E8h		2	Nombre de documents à lire.
0F2F1h		2	
0F304h		2	
0F306h		1	Lecteur utilisé par défaut (0 ~ 7). 0 = « A: », 1 = « B: », etc.
0F30Dh	RAWFLG	1	Indicateur de vérification. 0 = Vérification terminée ; Autre valeur = Vérification en cours.
0F30Eh		1	Indicateur du format de la date. 0 = AAMMJJ, 1= MMJJAA, 2= JJMMAA.
0F313h		1	Version de la Disk-ROM. 0 = Disk-ROM v.1.x, 2xH = Disk-ROM v.2.x.
0F323h	DISKVE	2	Adresse du pointeur de la routine de gestion des erreurs.
0F325h	BREAKV	2	Adresse du pointeur de la routine de gestion de CTRL-C
0F336h		1	Indicateur de la touche pressée. Contient 0FFh lorsqu'une touche est pressée. Contient 03h si CTRL et STOP sont pressées.
0F337h		1	Indicateur de la touche pressée. Contient le code ASCII de la touche pressée. Contient 03h si CTRL et STOP sont pressées.
0F338h		1	Drapeau pour indiquer la présence de l'horloge interne. 0 = horloge interne, Autre valeur = Pas d'horloge interne.
0F339h		7	Zone de travail de la puce de l'horloge interne.
0F340h		1	Etat du MSX-DOS. 0 = MSX-DOS installé.
0F341h	RAMAD0	1	Numéro du Slot qui contient la RAM principale du DOS de la page 0.
0F342h	RAMAD1	1	Numéro du Slot qui contient la RAM principale du DOS de la page 1.
0F343h	RAMAD2	1	Numéro du Slot qui contient la RAM principale du DOS de la page 2.
0F344h	RAMAD3	1	Numéro du Slot qui contient la RAM principale du DOS de la page 3.

0F345h		1	Nombre de Buffers disponible trouvé lors du calcul.
0F346h		1	Indicateur pour savoir si le système a démarré sous DOS ou pas (0). Par conséquent, CALL SYSTEM est possible si différent de 0.
0F347h	NMBDRV	1	Nombre de disque logique connectés (0 ~ 7).
0F348h	MASTERS	1	Numéro du Slot qui contient la Disk-ROM maitre.
0F349h	HIMSAV	2	Adresse de début de la zone de travail de la Disk-ROM.
0F34Bh		2	Adresse de début de la zone disponible en RAM principale du DOS. Le fichier COMMAND.COM se charge à cette adresse. (0000h par défaut.)
0F34Dh	SECBUF	2	Adresse du Buffer temporaire utilisé lors de lecture / écriture dans la FAT.
0F34Fh	BUFFER	2	Adresse du Buffer temporaire utilisé lors de lecture / écriture dans les secteurs de données.
0F351h	DIRBUF	2	Adresse du Buffer temporaire utilisé lors de lecture / écriture d'un secteur. Utilisé par les instructions DSKIS & DSKOS du Disk-Basic.
0F353h	FCBBASE	2	Adresse du FCB du fichier actuel.
0F355h	DPBLIST	2	Adresse du DPB du disque « A: ».
0F357h	"	2	Adresse du DPB du disque « B: ».
0F359h	"	2	Adresse du DPB du disque « C: ».
0F35Bh	"	2	Adresse du DPB du disque « D: ».
0F35Dh	"	2	Adresse du DPB du disque « E: ».
0F35Fh	"	2	Adresse du DPB du disque « F: ».
0F361h	"	2	Adresse du DPB du disque « G: ».
0F363h	"	2	Adresse du DPB du disque « H: ».
0F365h		3	Routine de lecture de l'état des Slots primaires. Sortie : A = État des Slots primaires.
0F368h	SETROM	3	Routine pour sélectionner la Disk-ROM sur la plage 04000h~7FFFh. (MSX-DOS seulement.)
0F36Bh	SETRAM	3	Routine pour sélectionner la Main-RAM sur la plage 04000h~7FFFh. (MSX-DOS seulement.)
0F36Eh	SLTMOV	3	Routine de copie de la Disk-ROM vers la Main-RAM. (Actif sous MSX-DOS seulement.) Entrée : HL = Source ; DE = Destination ; BC = Longueur.
0F371h	AUXINP	3	Routine de lecture du périphérique auxiliaire de la Disk-ROM. Sortie : A = 01Ah par défaut (^Z)
0F374h	AUXOUT	3	Routine d'écriture au périphérique auxiliaire de la Disk-ROM. Entrée : A = Octet à envoyer.
0F377h	BLDCHK	3	Routine BLOAD du Disk-Basic. (JP 0000h sous MSX-DOS)
0F37Ah	"	3	Routine BSAVE du Disk-Basic. (JP 0000h sous MSX-DOS)

0F37Dh	ROMBDOS	3	Routine BDOS de la Disk-ROM. Entrée : C = Numéro de la Routine à appeler.
0F459h		100	Contient la commande entrée sous MSX-DOS.
0F85Fh	MAXFIL	1	Nombre maximum de fichiers autorisé. Modifié par l'instruction MAXFILES du Disk-Basic.
0F860h	FILTAB	2	Pointeur sur l'adresse des données du fichier.
0F862h	NULBUF	2	Pointeur du Buffer des instructions SAVE et LOAD du Disk-Basic.
0F864h	PTRFIL	2	Pointeur sur les données du fichier sélectionné.
0F866h	RUNFLG	1	Indicateur d'exécution pour l'instruction LOAD. 0 si le programme a été exécuté par l'option R.
0F866h	FILNAM	11	Nom du fichier d'une instruction du Disk-Basic.
0F871h	FILNM2	11	Nom du second fichier des instructions du Disk-Basic (NAME, COPY, MOVE, etc).
0F87Ch	NLONLY	1	Indicateur différent de 0 lors d'un chargement de programme. Utilisé par les instructions BSAVE et CREATE.
0F87Dh	SAVEND	2	Adresse de fin de l'instruction BSAVE du Disk-Basic.
0FB21h	DRVINF	1	Nombre de lecteur physique contrôlé par la première Disk-ROM.
0FB22h	"	1	Numéro de Slot de la première Disk-ROM installée.
0FB23h	"	1	Nombre de lecteur physique contrôlé par la seconde Disk-ROM.
0FB24h	"	1	Numéro de Slot de la seconde Disk-ROM installée.
0FB25h	"	1	Nombre de lecteur physique contrôlé par la troisième Disk-ROM .
0FB26h	"	1	Numéro de Slot de la troisième Disk-ROM installée.
0FB27h	"	1	Nombre de lecteur physique contrôlé par la quatrième Disk-ROM.
0FB28h	"	1	Numéro de Slot de la quatrième Disk-ROM installée.
0FCBFh	SAVENT	2	Adresse de début spécifiée par l'instruction BSAVE du Disk-Basic.
0FFCFh	DISINT	5	Entrée : ? Note : Utilisé seulement par le MSX-DOS.
0FFD4h	ENAINIT	5	Entrée : ? Note : Utilisé seulement par le MSX-DOS.
0FFD9h		14	Zone de travail pour DISINT et ENAINIT.

8.2 Zone fixe de travail et des variables du MSX-DOS2 et du Disk-BASIC 2

Cette zone est assez différente de celle du MSX-DOS1. Seules quelques parties sont communes dont les Hooks.

Adresse	Nom	Long.	Fonction
00006h		2	Adresse de la fin de la zone réservée à l'utilisateur. Lorsqu'une commande du MSX-DOS (fichier COM) est lancée, elle se charge à l'adresse 0100h et sa taille peut aller jusqu'à l'adresse indiquée ici.
005Ch	FCBA		Cette zone contient le premier nom de fichier indiqué derrière la commande entrée. Le premier octet indique le numéro de lecteur.
006Ch	FCB2		Cette zone contient le deuxième nom de fichier indiqué derrière la commande entrée. Le premier octet indique le numéro de lecteur.
0080h	DMA		Zone DMA par défaut. Cette zone contient tous paramètres indiqués derrière la commande entrée. Le premier octet indique le nombre de caractères utilisés pour paramètres.
0F1D3h		3	Saut à la routine ?
0F1D6h		3	Saut à la routine ?
0F1D9h		3	Saut à la routine ?
0F1DCh		3	Saut à la routine ?
0F1DFh		3	Saut à la routine ?
0F1E2h		3	Saut à la routine ?
0F1E5h		3	Saut à la routine de gestion d'interruption. (appelé seulement lors du traitement des fonctions BDOS).
0F1E8h		3	Saut à la routine RDSLT du BIOS (000Ch). (appelé seulement lors du traitement des fonctions BDOS).
0F1EBh		3	Saut à la routine WRSLT du BIOS (0014h). (appelé seulement lors du traitement des fonctions BDOS).
0F1EEh		3	Saut à la routine CALSLT du BIOS (001Ch). (appelé seulement lors du traitement des fonctions BDOS).
0F1F1h		3	Saut à la routine ENASLT du BIOS (0024h). (appelé seulement lors du traitement des fonctions BDOS).
0F1F4h		3	Saut à la routine CALLF du BIOS (0030h). (appelé seulement lors du traitement des fonctions BDOS).
0F1F7h		3	Saut à la routine pour mettre en mode DOS. (pages 0 et 2 avec le système)
0F1FAh		3	Saut à la routine pour mettre en mode utilisateur (« User Mode »).
0F1FDh		3	Routine de sélection de la ROM du MSX-DOS 2 sur la plage 00000h~03FFFh.

0F200h	3	Saut à la routine qui alloue une page de Memory Mapper.
0F203h	3	Saut à la routine qui libère une page de Memory Mapper.
0F206h	3	Saut à la routine de lecture d'un octet sur une page de Memory Mapper. Entrée : HL = Adresse, A = Numéro de page.
0F209h	3	Saut à la routine d'écriture d'un octet sur une page de Memory Mapper. Entrée : HL = Adresse, A = Numéro de page. Sortie : E = Octet lu.
0F20Ch	3	Saut à la routine d'appel de routine se trouvant sur une page de Memory Mapper. Entrée : IX = adresse de la routine à appeler, IYh = page.
0F20Fh	3	Saut à la routine d'appel de routine se trouvant sur une page de Memory Mapper. Entrée : Octet après l'adresse de l'instruction CALL IX = numéro de page. Les deux octets suivants = adresse de la routine à appeler.
0F212h	3	Saut à la routine de sélection d'une page de Memory Mapper. Entrée : HL = adresse quelconque dans la plage mémoire sur laquelle la page sera sélectionnée, A = numéro de page à sélectionner.
0F215h	3	Saut à la routine indiquant la page de Memory Mapper actuelle. Entrée : HL = adresse quelconque dans la plage mémoire sur laquelle on cherche à connaître la page actuellement sélectionnée. Sortie : A = numéro de page.
0F218h	3	Routine de sélection d'une page de Memory Mapper sur la plage mémoire 0000h~03FFFh. Entrée : A = numéro de page.
0F21Bh	3	Routine d'écriture dans la variable système du numéro page du Memory Mapper actuellement sélectionnée sur la plage mémoire 00000h~03FFFh. Entrée : A = numéro de page.
0F21Eh	3	Routine de sélection d'une page de Memory Mapper sur la plage mémoire 04000h~07FFFh. Entrée : A = numéro de page.
0F221h	3	Routine d'écriture dans la variable système du numéro page du Memory Mapper actuellement sélectionnée sur la plage mémoire 04000h~07FFFh. Entrée : A = numéro de page.
0F224h	3	Routine de sélection d'une page de Memory Mapper sur la plage mémoire 8000h~0BFFFh. Entrée : A = numéro de page.
0F227h	3	Routine d'écriture dans la variable système du numéro page du Memory Mapper actuellement sélectionnée sur la plage

mémoire 08000h~0BFFFh.

Entrée : A = numéro de page.

0F22Ah		3	Pas de routine pour la plage mémoire 3. (saut à un RET.)
0F22Dh		3	Saut à la routine indiquant la page de Memory Mapper se trouvant dans la plage mémoire 3. Entrée : A = numéro de page.
0F23Ch		1	Numéro du lecteur logique actuelle.
0F23Dh		2	Adresse actuelle de la DMA. (0080h par défaut)
0F23Fh		4	Numéro du secteur actuel du disque.
0F243h		2	Pointeur à l'adresse du DPB du lecteur actuel.
0F245h		1	Numéro relatif du secteur dans le répertoire. Utilisé par les routines SEARCH FIRST et NEXT. (Ajouter le numéro du premier secteur du répertoire pour connaître le numéro de secteur réel.)
0F246h		1	Numéro du lecteur dont le répertoire a été lu. (0 = « A: », 1 = « B: », etc)
0F247h		1	Numéro du lecteur par défaut. (0 = « A: », 1 = « B: », etc)
0F24Fh		3	Hook appelé avant ?
0F252h	H.BDOS	3	Hook appelé avant l'exécution d'une routine BDOS. Page 0 - map bloc (F2D0h) ; Page 2 - map block (F2CFh)
0F255h		3	Hook appelé avant ?
0F258h		3	Hook appelé avant ?
0F261h	H.	3	Hook appelé au début de la fonction 02h du BDOS.
0F283h		2	Adresse de la TPA. (MSX-DOS 2.33)
0F2B6h		1	Indicateur disponibles sous MSX-DOS 2.33 bit 0~2 = Réservés, bit 3 = VDP rapide OFF, bit 4 = TPA propre OFF (0F2B3h = adresse de la TPA), bit 5 = RESET ON, bit 6 = BUSRESET OFF, bit 7 = REBOOT ON.
0F2B9h		1	Compteur ?
0F2C0h		5	Second Hook de la routine d'interruption (utilisé par la Disk-ROM).
0F2C5h		2	Adresse de la table de correspondance.
0F2C7h		1	Page du Memory Mapper sélectionnée sur la plage 0000h~3FFFh.
0F2C8h		1	Page du Memory Mapper sélectionnée sur la plage

		4000h~7FFFh.
0F2C9h	1	Page du Memory Mapper sélectionnée sur la plage 8000h~BFFFh.
0F2CAh	1	Page du Memory Mapper sélectionnée sur la plage C000h~FFFFh.
0F2CBh	1	Le contenu de F2C7h est copié ici lors de l'exécution de la routine BDOS.
0F2CCh	1	Le contenu de F2C8h est copié ici lors de l'exécution de la routine BDOS.
0F2CDh	1	Le contenu de F2C9h est copié ici lors de l'exécution de la routine BDOS.
0F2CEh	1	Le contenu de F2CAh est copié ici lors de l'exécution de la routine BDOS.
0F2CFh	1	Numéro de la dernière page disponible dans le Memory Mapper primaire. Buffer utilisé lors d'un appel à une routines BDOS sur la page 2. (zone de travail temporaire.)
0F2D0h	1	Numéro de la dernière page disponible dans le Memory Mapper primaire. Buffer utilisé lors d'un appel à une routines BDOS sur la page 0. (zone de travail temporaire.)
0F2D1h	2	Adresse
0F2D3h	2	Adresse
0F2D5h	5	Deuxième Hook de la routine FCALL (FFCAh) du BIOS étendu.
0F2DAh	4	Adresse de la routine d'appel des fonctions BDOS de la deuxième ROM.
0F2DEh	4	Adresse de la routine d'appel des fonctions BDOS de la ROM du DOS2. actuellement sélectionnée
0F2E0h	1	Numéro des Slots primaires sélectionnés sur chaque plage.
0F2E6h	2	Buffer utilisé pour registre de stockage temporaire IX
0F2E8h	2	Buffer utilisé pour le stockage temporaire du registre SP.
0F2EAh	1	Statut des slots primaires après l'exécution d'une fonction de BDOS.
0F2EBh	1	Statut des slots secondaires après l'exécution d'une fonction de BDOS.
0F2ECh	1	Indicateur pour vérifier l'état du disque. (0 = OFF, autre valeur = ON.)
0F2FBh	2	Pointeur vers un buffer temporaire utilisé pendant l'interprétation d'un code d'erreur.
0F2FDh	1	Adresse du Buffer temporaire de la fonction « Explain Error Code ».
0F2FEh	2	Pointeur du haut la pile du DOS2.

0F300h		13	?
0F30Dh		1	Vérification du disque (00h = OFF, 01h = ON.)
0F313h		1	Version de la Disk-ROM. 0 = Disk-ROM v.1.x, 2xh = Disk-ROM v.2.x.
0F314h		1	Page du Memory Mapper sur la plage 0000h~3FFFh.
0F315h		1	Page du Memory Mapper sur la plage 4000h~7FFFh.
0F316h		1	Page du Memory Mapper sur la plage 8000h~BFFFh.
0F317h		1	Page du Memory Mapper sur la plage C000h~FFFFh.
0F33Fh		2	Adresse de ?.
0F341h		1	Numéro du Slot qui contient la RAM principale du DOS de la plage 0.
0F342h		1	Numéro du Slot qui contient la RAM principale du DOS de la plage 1.
0F343h		1	Numéro du Slot qui contient la RAM principale du DOS de la plage 2.
0F344h		1	Numéro du Slot qui contient la RAM principale du DOS de la plage 3.
0F345h		1	Nombre de Buffers disponible trouvé lors du calcul.
0F346h		1	Indicateur pour savoir si le système a démarré sous DOS ou pas (0). Par conséquent, CALL SYSTEM est possible si différent de 0.
0F347h	NMBDRV	1	Nombre de disque logique connectés (0 ~ 7).
0F348h		1	Numéro du Slot qui contient la Disk-ROM.
0F349h		2	Adresse de début de la zone de travail de la Disk-ROM.
0F34Bh		2	Adresse de début de la zone disponible en RAM principale du DOS. Le fichier COMMAND.COM se charge à cette adresse. (0000h par défaut.)
0F34Dh		2	Adresse du Buffer temporaire utilisé lors de lecture / écriture dans la FAT.
0F34Fh		2	Adresse du Buffer temporaire utilisé lors de lecture / écriture dans les secteurs de données.
0F351h		2	Adresse du Buffer temporaire utilisé lors de lecture / écriture d'un secteur. Utilisé par les instructions DSKI\$ & DSKO du Disk-Basic.
0F353h		2	Adresse du FCB du fichier actuel.
0F355h	DPBLST	2	Adresse du DPB du disque « A: ».
0F357h	"	2	Adresse du DPB du disque « B: ».
0F359h	"	2	Adresse du DPB du disque « C: ».

0F35Bh	"	2	Adresse du DPB du disque « D: ».
0F35Dh	"	2	Adresse du DPB du disque « E: ».
0F35Fh	"	2	Adresse du DPB du disque « F: ».
0F361h	"	2	Adresse du DPB du disque « G: ».
0F363h	"	2	Adresse du DPB du disque « H: ».
0F365h		3	Routine de lecture de l'état des Slots primaires. Sortie : A = État des Slots primaires.
0F377h		3	Routine d'appel des fonctions BDOS en page 0.
0F37Ah		3	Deuxième routine d'appel des fonctions BDOS en page 0.
0F37Dh		3	Routine BDOS de la Disk-ROM (du MSX-DOS2). Entrée : C = Numéro de la Routine à appeler.
0F459h		100	Contient la dernière commande entrée sous MSX-DOS.
0F85Fh	MAXFIL	1	Nombre maximum de fichiers autorisé. Modifié par l'instruction MAXFILES du Disk-Basic.
0F860h	FILTAB	2	Pointeur sur l'adresse des données du fichier.
0F862h	NULBUF	2	Pointeur du Buffer des instructions SAVE et LOAD du Disk-Basic.
0F864h	PTRFIL	2	Pointeur sur les données du fichier sélectionné.
0F866h	RUNFLG	1	Indicateur d'exécution pour l'instruction LOAD. 0 si le programme a été exécuté par l'option R.
0F866h	FILNAM	11	Nom du fichier d'une instruction du Disk-Basic.
0F871h	FILNM2	11	Nom du second fichier des instructions du Disk-Basic (NAME, COPY, MOVE, etc).
0F87Ch	NLONLY	1	Indicateur différent de 0 lors d'un chargement de programme. Utilisé par les instructions BSAVE et CREATE.
0F87Dh	SAVEND	2	Adresse de fin de l'instruction BSAVE du Disk-Basic.
0FB21h	DRVINV	1	Nombre de lecteur physique contrôlé par la première Disk-ROM.
0FB22h	"	1	Numéro de Slot de la première Disk-ROM installée.
0FB23h	"	1	Nombre de lecteur physique contrôlé par la seconde Disk-ROM.
0FB24h	"	1	Numéro de Slot de la seconde Disk-ROM installée.
0FB25h	"	1	Nombre de lecteur physique contrôlé par la troisième Disk-ROM .
0FB26h	"	1	Numéro de Slot de la troisième Disk-ROM installée.
0FB27h	"	1	Nombre de lecteur physique contrôlé par la quatrième Disk-

ROM.

0FB28h	"	1	Numéro de Slot de la quatrième Disk-ROM installée.
0FB29h		1	Numéro de Slot ?
0FCBFh	SAVENT	2	Adresse de début spécifiée par l'instruction BSAVE du Disk-Basic.
0FFCAh	EXTBIO	5	Appel à une routine du Bios d'une extension. Entrée : D = Identifiant de l'extension. 0 = Tous, 4 = Memory Mapper, 8 =MSX-Modem ou RS-232C, 10 = MSX-Audio, 16 = MSX-JE, 17 = Kanji driver, 34 = UNAPI (Konamiman), 77 = Memman, 255 = Système. E = Numéro de la routine à appeler. Sortie : Dépend de la routine appelée. Note : Cette routine est présente si le bit 0 de HOKVLD (0FB20h) est à 1. (Voir le paragraphe sur le Bios étendu pour plus de précision)
0FFCFh	DISINT	5	
0FFD4h	ENAIN	5	
0FFD9h		14	Zone de travail pour DISINT / ENAIN.

Annexes

A - Les BIOS

Voici la liste de toutes les routines des Bios classées par ordre alphabétique :

<u>Nom</u>	<u>Adresse</u>	<u>Slot</u>	<u>Nom</u>	<u>Adresse</u>	<u>Slot</u>
ATRSCN	00071h	Sub-ROM	CHGMOD	0005Fh	Main-ROM
BASE	00169h	Sub-ROM	CHGMOD	000D1h	Sub-ROM
BASEF	0016Dh	Sub-ROM	CHGSND	00135h	Main-ROM
BASRVN	0002Bh	Main-ROM	CHKNEW	00165h	Main-ROM
BEEP	000C0h	Main-ROM	CHKSLZ	00162h	Main-ROM
BEEP	0017Dh	Sub-ROM	CHPUT	000A2h	Main-ROM
BIGFIL	0016Bh	Main-ROM	CHRGTR	00010h	Main-ROM
BLTDM	001A9h	Sub-ROM	CHSNS	0009Ch	Main-ROM
BLTDV	001A1h	Sub-ROM	CKCNTC	000BDh	Main-ROM
BLTMD	001A5h	Sub-ROM	CLRSPR	00069h	Main-ROM
BLTMV	00199h	Sub-ROM	CLRSPR	000F5h	Sub-ROM
BLTVM	00195h	Sub-ROM	CLRTXT	00119h	Sub-ROM
BLTVD	0019Dh	Sub-ROM	CLS	000C3h	Main-ROM
BLTVV	00191h	Sub-ROM	CLS	00115h	Sub-ROM
BOXLIN	00081h	Sub-ROM	CNVCHR	000ABh	Main-ROM
BREAKX	000B7h	Main-ROM	COLOR	00155h	Sub-ROM
CALATR	00087h	Main-ROM	DCOMPR	00020h	Main-ROM
CALATR	000FDh	Sub-ROM	DELLNO	00121h	Sub-ROM
CALBAS	00159h	Main-ROM	DISSCR	00041h	Main-ROM
CALLF	00030h	Main-ROM	DOBOXF	00079h	Sub-ROM
CALPAT	00084h	Main-ROM	DOGRPH	00085h	Sub-ROM
CALPAT	000F9h	Sub-ROM	DOLINE	0007Dh	Sub-ROM
CALSLT	0001Ch	Main-ROM	DOWNC	00108h	Main-ROM
CGTABL	00004h	Main-ROM	DOWNC	000B5h	Sub-ROM
CHGCAP	00132h	Main-ROM	DSPFNK	000CFh	Main-ROM
CHGCPU	00180h	Main-ROM	DSPFNK	0011Dh	Sub-ROM
CHGCLR	00062h	Main-ROM	ENASCR	00044h	Main-ROM
CHGCLR	00111h	Sub-ROM	ENASLT	00024h	Main-ROM
CHGET	0009Fh	Main-ROM	EOL	00168h	Main-ROM
CHGMDP	001B5h	Sub-ROM	ERAFNK	000CCh	Main-ROM

EXTROM	0015Fh	Main-ROM	ISFLIO	0014Ah	Main-ROM
FETCHC	00114h	Main-ROM	KEYINT	00038h	Main-ROM
FILVRM	00056h	Main-ROM	KILBUF	00156h	Main-ROM
FNKSB	000C9h	Main-ROM	KNJPRT	001BDh	Sub-ROM
FORMAT	00147h	Main-ROM	KYKLOK	00135h	Sub-ROM
GETCPU	00183h	Main-ROM	LDIRMV	00059h	Main-ROM
GETPAT	00105h	Sub-ROM	LDIRVM	0005Ch	Main-ROM
GETPLT	00149h	Sub-ROM	LEFTC	000FFh	Main-ROM
GETPUT	001B1h	Sub-ROM	LEFTC	000ADh	Sub-ROM
GETVCP	00150h	Main-ROM	LFTQ	000F6h	Main-ROM
GETVC2	00153h	Main-ROM	LPTOUT	000A5h	Main-ROM
GETYPR	00028h	Main-ROM	LPTSTT	000A8h	Main-ROM
GICINI	00090h	Main-ROM	MAPXYC	00111h	Main-ROM
GLINE	00075h	Sub-ROM	MAPXYC	00091h	Sub-ROM
GRPPRT	0008Dh	Main-ROM	MSXVER	0002Dh	Main-ROM
GRPPRT	00089h	Sub-ROM	NEWPAD	001ADh	Sub-ROM
GSPSIZ	0008Ah	Main-ROM	NMI	00066h	Main-ROM
GSPSIZ	00101h	Sub-ROM	NRDVRM	00174h	Main-ROM
GTASPC	00126h	Main-ROM	NSETCX	00123h	Main-ROM
GTPAD	000DBh	Main-ROM	NSETRD	0016Eh	Main-ROM
GTPDL	000DEh	Main-ROM	NSTWRT	00171h	Main-ROM
GTSTCK	000D5h	Main-ROM	NVBXFL	000CDh	Sub-ROM
GTTRIG	000D8h	Main-ROM	NVBXLN	000C9h	Sub-ROM
INIFNK	0003Eh	Main-ROM	NWRVRM	00177h	Main-ROM
INIGRP	00072h	Main-ROM	OUTLDP	0014Dh	Main-ROM
INIGRP	000DDh	Sub-ROM	OUTDO	00018h	Main-ROM
INIMLT	00075h	Main-ROM	PAINT	00069h	Sub-ROM
INIMLT	000E1h	Sub-ROM	PCMPLY	00186H	Main-ROM
INIPLT	00141h	Sub-ROM	PCMREC	00189H	Main-ROM
INITIO	0003Bh	Main-ROM	PNTINI	00129h	Main-ROM
INITXT	0006Ch	Main-ROM	PINLIN	000AEh	Main-ROM
INITXT	000D5h	Sub-ROM	POSIT	000C6h	Main-ROM
INIT32	0006Fh	Main-ROM	PROMPT	00181h	Sub-ROM
INIT32	000D9h	Sub-ROM	PSET	0006Dh	Sub-ROM
INLIN	000B1h	Main-ROM	PUTCHR	00139h	Sub-ROM
INSLN0	00125h	Sub-ROM	PUTQ	000F9h	Main-ROM
ISCNTC	000BAh	Main-ROM	PUTSPR	00151h	Sub-ROM

PUTVRM	00129h	Sub-ROM	SETSCR	00189h	Sub-ROM
PYSDIO	00144h	Main-ROM	SETTXT	00078h	Main-ROM
QINLIN	000B4h	Main-ROM	SETTXT	000E5h	Sub-ROM
RDPSG	00096h	Main-ROM	SETT32	0007Bh	Main-ROM
RDRES	0017Ah	Main-ROM	SETT32	000E9h	Sub-ROM
RDSLTL	0000Ch	Main-ROM	SETWRT	00053h	Main-ROM
RDVDP	0013Eh	Main-ROM	SNSMAT	00141h	Main-ROM
RDVRM	0004Ah	Main-ROM	STARUP	00000h	Main-ROM
RDVRM	0010Dh	Sub-ROM	STMOTR	000F3h	Main-ROM
READC	0011Dh	Main-ROM	STOREC	00117h	Main-ROM
READC	00095h	Sub-ROM	STRTMS	00099h	Main-ROM
REDCLK	001F5h	Sub-ROM	SUBROM	0015Ch	Main-ROM
RIGHTC	000FCh	Main-ROM	SYNCHR	00008h	Main-ROM
RIGHTC	000A5h	Sub-ROM	TAPIN	000E4h	Main-ROM
RSLREG	00138h	Main-ROM	TAPIOF	000E7h	Main-ROM
RSTPLT	00145h	Sub-ROM	TAPION	000E1h	Main-ROM
SCALXY	0010Eh	Main-ROM	TAPOOF	000F0h	Main-ROM
SCALXY	0008Dh	Sub-ROM	TAPOON	000EAh	Main-ROM
SCANL	0012Fh	Main-ROM	TAPOUT	000EDh	Main-ROM
SCANL	000C5h	Sub-ROM	TDOWNC	0010Bh	Main-ROM
SCANR	0012Ch	Main-ROM	TDOWNC	000B1h	Sub-ROM
SCANR	000C1h	Sub-ROM	TLEFTC	000A9h	Sub-ROM
SCOPY	0018Dh	Sub-ROM	TOTEXT	000D2h	Main-ROM
SCREEN	00159h	Sub-ROM	TRIGHT	000A1h	Sub-ROM
SDFSCR	00185h	Sub-ROM	TUPC	00105h	Main-ROM
SETATR	0011Ah	Main-ROM	TUPC	000B9h	Sub-ROM
SETATR	00099h	Sub-ROM	UPC	00102h	Main-ROM
SETC	00120h	Main-ROM	UPC	000BDh	Sub-ROM
SETC	0009Dh	Sub-ROM	VDP	00161h	Sub-ROM
SETGRP	0007Eh	Main-ROM	VDPF	00165h	Sub-ROM
SETGRP	000EDh	Sub-ROM	VDPSTA	00131h	Sub-ROM
SETMLT	00081h	Main-ROM	VDP.DR	00006h	Main-ROM
SETMLT	000F1h	Sub-ROM	VDP.DW	00007h	Main-ROM
SETPAG	0013Dh	Sub-ROM	VPEEK	00175h	Sub-ROM
SETPLT	0014Dh	Sub-ROM	VPOKE	00171h	Sub-ROM
SETRD	00050h	Main-ROM	WIDTHS	0015Dh	Sub-ROM
SETS	00179h	Sub-ROM	WRRES	0017Dh	Main-ROM

WRTCLK	001F9h	Sub-ROM	WRTVDP	0012Dh	Sub-ROM
WRTPSG	00093h	Main-ROM	WRTVRM	0004Dh	Main-ROM
WRTSLT	00014h	Main-ROM	WRTVRM	00109h	Main-ROM
WRTVDP	00047h	Main-ROM	WSLREG	0013Bh	Main-ROM

B - Les Variables Système

Liste des variables système dans l'ordre alphabétique :

<u>Nom</u>	<u>Adresse</u>	<u>Longueur</u>	<u>Type</u>	<u>Nom</u>	<u>Adresse</u>	<u>Longueur</u>	<u>Type</u>
ACPAGE	0FAF6h	1	MSX2	CLOC	0F92Ah	2	MSX1
ARG	0F847h	16	MSX1	CMASK	0F92Ch	1	MSX1
ARYTAB	0F6C4h	2	MSX1	CNPNTS	0F936h	2	MSX1
ARYTA2	0F7B5h	2	MSX1	CNSDFG	0F3DEh	1	MSX1
ASPCT1	0F40Bh	2	MSX1	CODSAV	0FBCCh	1	MSX1
ASPCT2	0F40Dh	2	MSX1	CONLO	0F66Ah	8	MSX1
ASPECT	0F931h	2	MSX1	CONSAV	0F668h	1	MSX1
ATRBAS	0F928h	2	MSX1	CONTXT	0F666h	2	MSX1
ATRBYT	0F3F2h	1	MSX1	CONTYP	0F669h	1	MSX1
AUTFLG	0F6AAh	1	MSX1	CPCNT	0F939h	2	MSX1
AUTINC	0F6ADh	2	MSX1	CPCNT8	0F93Bh	2	MSX1
AUTLIN	0F6ABh	2	MSX1	CPLOTF	0F938h	1	MSX1
AVCSAV	0FAF7h	1	MSX2	CRCSUM	0h93Dh	2	MSX1
BAKCLR	0F3EAh	1	MSX1	CRTCNT	0F3B1h	1	MSX1
BASROM	0FBB1h	1	MSX1	CS120	0F3FCh	10	MSX1
BDRCLR	0F3EBh	1	MSX1	CSAVEA	0F942h	2	MSX1
BOTTOM	0FC48h	2	MSX1	CSAVEM	0F944h	1	MSX1
BRDATR	0FCB2h	1	MSX1	CSCLXY	0F941h	1	MSX1
BUF	0F55Eh	256	MSX1	CSRSW	0FCA9h	1	MSX1
BUFEND	0FC18h	0	MSX1	CSRX	0F3DDh	1	MSX1
BUFMIN	0F55Dh	1	MSX1	XSRV	0F3DCh	1	MSX1
CAPST	0FCABh	1	MSX1	CSTCNT	0F93Fh	2	MSX1
CASPRV	0FCB1h	1	MSX1	CSTYLE	0FCAAh	1	MSX1
CENCNT	0F933h	1	MSX1	CURLIN	0F41Ch	2	MSX1
CGPBAS	0F924h	2	MSX1	CXOFF	0F945h	2	MSX1
CGPNT	0F91Fh	2	MSX1	CYOFF	0F947h	2	MSX1
CHRCNT	0FAF9h	3	MSX2	DAC	0F7F6h	16	MSX1
CLIKFL	0FBD9h	1	MSX1	DATLIN	0F6A3h	2	MSX1
CLIKSW	0F3DBh	1	MSX1	DATPTR	0F6C8h	2	MSX1
CLINEF	0F935h	1	MSX1	DECCNT	0F7F4h	1	MSX1
CLMSLT	0F3B2h	1	MSX1	DECTMP	0F7F0h	2	MSX1

DECTM2	0F7F2h	2	MSX1	FUNACT	0F7BAh	2	MSX1
DEFTBL	0F6CAh	26	MSX1	GETPNT	0F3FAh	2	MSX1
DEVICE	0FD99h	1	MSX1	GRPACX	0FCB7h	2	MSX1
DIMFLG	0F662h	1	MSX1	GRPACY	0FCB9h	2	MSX1
DORES	0F664h	1	MSX1	GRPATR	0F3CDh	2	MSX1
DONUM	0F665h	1	MSX1	GRPCGP	0F3CBh	2	MSX1
DOT	0F6B5h	2	MSX1	GRPCOL	0F3C9h	2	MSX1
DPPAGE	0FAF5h	1	MSX2	GRPHED	0FCA6h	1	MSX1
DRWANG	0FCBDh	1	MSX1	GRPNAM	0F3C7h	2	MSX1
DRWFLG	0FCBBh	1	MSX1	GRPPAT	0F3CFh	2	MSX1
DRWSCL	0FCBCh	1	MSX1	GXPOS	0FCB3h	2	MSX1
DSCPTR	0F699h	interne	MSX1	GYPOS	0FCB5h	2	MSX1
DSCTMP	0F698h	3	MSX1	HEADER	0F40Ah	1	MSX1
ENDBUF	0F660h	1	MSX1	HIGH	0F408h	2	MSX1
ENDFOR	0F6A1h	2	MSX1	HIMEM	0FC4Ah	2	MSX1
ENDPRG	0F40Fh	5	MSX1	HOLD	0F83Eh	8	MSX1
ENSTOP	0FBB0h	1	MSX1	HOLD2	0F836h	8	MSX1
ERRFLG	0F414h	1	MSX1	HIGH	0F408h	2	MSX1
ERRLIN	0F6B3h	2	MSX1	HOLD8	0F806h	48	MSX1
ERRTXT	0F6B7h	2	MSX1	INSFLG	0FCA8h	1	MSX1
ESCCNT	0FCA7h	1	MSX1	INTCNT	0FCA2h	2	MSX1
EXBRSA	0FAF8h	1	MSX2	INTFLG	0FC9Bh	1	MSX1
EXPTBL	0FCC1h	4	MSX1	INTVAL	0FCA0h	2	MSX1
FACLO	0F7F8h	interne	MSX1	JIFFY	0FC9Eh	2	MSX1
FBuffer	0F7C5h	43	MSX1	KANAMD	0FCADh	1	MSX1
FILNAM	0F866h	11	MSX1	KANAST	0FCACH	1	MSX1
FILNM2	0F871h	11	MSX1	KBUF	0F41Fh	316	MSX1
FILTAB	0F860h	2	MSX1	KEYBUF	0FBF0h	40	MSX1
FLBMEM	0FCAEh	1	MSX1	LFPROG	0F954h	1	MSX1
FLGINP	0F6A6h	1	MSX1	LINL32	0F3AFh	1	MSX1
FKNFLG	0FBCEh	10	MSX1	LINL40	0F3AEh	1	MSX1
FNKSTR	0F87Fh	160	MSX1	LINLEN	0F3B0h	1	MSX1
FNKSWI	0FBCDh	1	MSX1	LINTTB	0FBB2h	24	MSX1
FORCLR	0F3E9h	1	MSX1	LINWRK	0FC18h	40	MSX1
FRCNEW	0F3F5h	1	MSX1	LOGOPR	0FB02h	1	MSX2
FRETOP	0F69Bh	2	MSX1	LOHADR	0F94Bh	2	MSX1
FSTPOS	0FBCAh	2	MSX1	LOHCNT	0F94Dh	2	MSX1

LOHDIR	0F94Ah	1	MSX1	OPRTYP	0F664h	0	MSX1
LOHMSK	0F949h	1	MSX1	PADX	0FC9Dh	1	MSX1
LOW	0F406h	2	MSX1	PADY	0FC9Ch	1	MSX1
LOWLIM	0FCA4h	1	MSX1	PARM1	0F6E8h	100	MSX1
LPTPOS	0F415h	1	MSX1	PARM2	0F750h	100	MSX1
MAXDEL	0F92Fh	2	MSX1	PATBAS	0F926h	2	MSX1
MAXFIL	0F85Fh	1	MSX1	PATWRK	0FC40h	8	MSX1
MAXUPD	0F3ECh	3	MSX1	PDIREC	0F953h	1	MSX1
MCLFLG	0F958h	1	MSX1	PLYCNT	0FB40h	1	MSX1
MCLLEN	0FB3Bh	1	MSX1	PRMFLG	0F7B4h	1	MSX1
MCLPTR	0FB3Ch	2	MSX1	PRMLEN	0F6E6h	2	MSX1
MCLTAB	0F956h	2	MSX1	PRMLN2	0F74Eh	2	MSX1
MEMSIZ	0F672h	2	MSX1	PRMPRV	0F74Ch	2	MSX1
MINDEL	0F92Dh	2	MSX1	PRMSTK	0F6E4h	2	MSX1
MINUPD	0F3EFh	3	MSX1	PROCNM	0FD89h	16	MSX1
MLTATR	0F3D7h	2	MSX1	PRSCNT	0FB35h	1	MSX1
MLTCGP	0F3D5h	2	MSX1	PTRFIL	0F864h	2	MSX1
MLTCOL	0F3D3h	2	MSX1	PTRFLG	0F6A9h	1	MSX1
MLTNAM	0F3D1h	2	MSX1	PUTPNT	0F3F8h	2	MSX1
MLTPAT	0F3D9h	2	MSX1	QUEBAK	0F971h	4	MSX1
MNROM	0FCC1h	1	MSX2	QUETAB	0F959h	24	MSX1
MODE	0FAFCh	1	MSX2	QUEUEN	0FB3Eh	1	MSX1
MOVCNT	0F951h	2	MSX1	QUEUES	0F3F3h	2	MSX1
MUSICF	0FB3Fh	1	MSX1	RAWPRT	0F418h	1	MSX1
NAMBAS	0F922h	2	MSX1	REPCNT	0F3F7h	1	MSX1
NEWKEY	0FBE5h	11	MSX1	REQSTP	0FC6Ah	1	MSX1
NLONLY	0F87Ch	1	MSX1	RG0SAV	0F3DFh	1	MSX1
NOFUNS	0F7B7h	1	MSX1	RG1SAV	0F3E0h	1	MSX1
NTMSXP	0F417h	1	MSX1	RG2SAV	0F3E1h	1	MSX1
NULBUF	0F862h	2	MSX1	RG3SAV	0F3E2h	1	MSX1
OLDKEY	0FBDAh	11	MSX1	RG4SAV	0F3E3h	1	MSX1
OLDLIN	0F6BEh	2	MSX1	RG5SAV	0F3E4h	1	MSX1
OLDSCR	0FCB0h	1	MSX1	RG6SAV	0F3E5h	1	MSX1
OLDTXT	0F6C0h	2	MSX1	RG7SAV	0F3E6h	1	MSX1
ONEFLG	0F6BBh	1	MSX1	ROMA	0FAFAh	2	MSX2
ONELIN	0F6B9h	2	MSX1	RNDX	0F857h	8	MSX1
ONGSBF	0FBD8h	1	MSX1	RSFCB	0FB04h	1	MSX2

RSIQLN	0FB06h	1	MSX2	TOCNT	0FB03h	1	MSX2
RS2IQ	0FAF5h	64	MSX1	TRCFLG	0F7C4h	1	MSX1
RTPROG	0F955h	1	MSX1	TRGFLG	0F3E8h	1	MSX1
RTYCNT	0FC9Ah	1	MSX1	TRPTBL	0FC4Ch	78	MSX1
RUNBNF	0FCBEh	1	MSX1	TTYPOS	0F661h	1	MSX1
RUNFLG	0F866h	0	MSX1	TXTATR	0F3B9h	2	MSX1
SAVEND	0F87Dh	2	MSX1	TXTCGP	0F3B7h	2	MSX1
SAVENT	0FCBFh	2	MSX1	TXTCOL	0F3B5h	2	MSX1
SAVSP	0FB36h	2	MSX1	TXTNAM	0F3B3h	2	MSX1
SAVSTK	0F6B1h	2	MSX1	TXTPAT	0F3BBh	2	MSX1
SAVTXT	0F6AFh	2	MSX1	TXTTAB	0F676h	2	MSX1
SAVVOL	0FB39h	2	MSX1	T32ATR	0F3C3h	2	MSX1
SCNCNT	0F3F6h	1	MSX1	T32CGP	0F3C1h	2	MSX1
SCRMOD	0FCAFh	1	MSX1	T32COL	0F3BFh	2	MSX1
SFTKEY	0FBEBh	interne	MSX1	T32NAM	0F3BDh	2	MSX1
SKPCNT	0F94Fh	2	MSX1	T32PAT	0F3C5h	2	MSX1
SLTATR	0FCC9h	64	MSX1	USRTAB	0F39Ah	20	MSX1
SLTTBL	0FCC5h	4	MSX1	VALTYP	0F663h	1	MSX1
SLTWRK	0FD09h	128	MSX1	VARTAB	0F6C2h	2	MSX1
STATFL	0F3E7h	1	MSX1	VCBA	0FB41h	37	MSX1
STKTOP	0F674h	2	MSX1	VCBB	0FB66h	37	MSX1
STREND	0F6C6h	2	MSX1	VCBC	0FB8Bh	37	MSX1
SUBFLG	0F6A5h	1	MSX1	VLZADR	0F419h	2	MSX1
SWPTMP	0F7BCh	8	MSX1	VLZDAT	0F41Bh	1	MSX1
TEMP	0F6A7h	2	MSX1	VOICAQ	0F975h	128	MSX1
TEMP2	0F6BCh	2	MSX1	VOICBQ	0F9F5h	128	MSX1
TEMP3	0F69Dh	2	MSX1	VOICBC	0FA75h	128	MSX1
TEMP8	0F69Fh	2	MSX1	VOICEN	0FB38h	1	MSX1
TEMP9	0F7B9h	2	MSX1	WINWID	0FCA5h	1	MSX1
TEMPPT	0F678h	2	MSX1	XSAVE	0FAFEh	2	MSX2
TEMPST	0F67Ah	30	MSX1	YSAVE	0FB00h	2	MSX2

0 : Adresse partagée par plusieurs variables.

C - Les Hooks

Voici la liste des Hooks classés dans l'ordre alphabétique :

Nom	Adresse	Nom	Adresse	Nom	Adresse	Nom	Adresse
H.ATTR	0FE1Ch	H.DSKI	0FE17h	H.GETP	0FE4Eh	H.MKSS\$	0FE35h
H.BAKU	0FEADh	H.DSKO	0FDEFh	H.GONE	0FF43h	H.NAME	0FDF9h
H.BINL	0FE76h	H.DSPC	0FDA9h	H.INDS	0FEA8h	H.NEWS	0FF3Eh
H.BINS	0FE71h	H.DSPF	0FDB3h	H.INIP	0FDC7h	H.NMI	0FDD6h
H.BUFL	0FF8Eh	H.EOF	0FEA3h	H.INLI	0FDE5h	H.NODE	0FEB7h
H.CHGE	0FDC2h	H.ERAC	0FDAEh	H.IPL	0FE03h	H.NOFO	0FE58h
H.CHPU	0FDA4h	H.ERAF	0FDB8h	H.ISFL	0FEDFh	H.NOTR	0FF34h
H.CHRG	0FF48h	H.ERRF	0FF02h	H.ISMI	0FF7Fh	H.NTFL	0FE62h
H.CLEA	0FED0h	H.ERRO	0FFB1h	H.ISRE	0FF2Ah	H.NTFN	0FF2Fh
H.CMD	0FE0Dh	H.ERRP	0FEFDh	H.KEYC	0FDCCh	H.NTPL	0FF6Bh
H.COMP	0FF57h	H.EVAL	0FF70h	H.KEYI	0FD9Ah	H.NULO	0FE5Dh
H.COPY	0FE08h	H.FIEL	0FE2Bh	H.KILL	0FDFEh	H.OKNO	0FF75h
H.CRDO	0FEE9h	H.FILE	0FE7Bh	H.KYEA	0FDD1h	H.ONGO	0FDEAh
H.CRUN	0FF20h	H.FILO	0FE85h	H.LIST	0FF89h	H.OUTD	0FEE4h
H.CRUS	0FF25h	H.FINE	0FF1Bh	H.LOC	0FE99h	H.PARD	0FEB2h
H.CVD	0FE49h	H.FING	0FF7Ah	H.LOF	0FE9Eh	H.PHYD	0FFA7h
H.CVI	0FE3Fh	H.FINI	0FF16h	H.LOPD	0FED5h	H.PINL	0FDDb
H.CVS	0FE44h	H.FINP	0FF5Ch	H.LPTO	0FFB6h	H.PLAY	0FFC5h
H.DEVN	0FEC1h	H.FORM	0FFACh	H.LPTS	0FFBBh	H.POSD	0FEBCh
H.DGET	0FFE8h	H.FPOS	0FEA8h	H.LSET	0FE21h	H.PRGE	0FEF8h
H.DIRD	0FF11h	H.FRET	0FF9Dh	H.MAIN	0FF0Ch	H.PRTF	0FF52h
H.DOGR	0FEF3h	H.FRME	0FF66h	H.MERG	0FE67h	H.PTRG	0FFA2h
H.DSKC	0FEEh	H.FRQI	0FF93h	H.MKDS\$	0FE3Ah	H.QINL	0FDE0h
H.DSKF	0FE12h	H.GEND	0FEC6h	H.MKI\$	0FE30h	H.READ	0FF07h
H.RETU	0FF4Dh	H.SAVD	0FE94h	H.SETF	0FE53h	H.TIMI	0FD9Fh
H.RSET	0FE26h	H.SAVE	0FE6Ch	H.SETS	0FDF4h	H.TOTE	0FDBDh
H.RSLF	0FE8Fh	H.SCNE	0FF98h	H.SNGF	0FF39h	H.TRMN	0FF61h
H.RUNC	0FECBh	H.SCRE	0FFC0h	H.STKE	0FEDAh	H.WIDT	0FF84h

Registre 36 :	DX7	DX6	DX5	DX4	DX3	DX2	DX1	DX0
Registre 37 :	0	0	0	0	0	0	0	DX8
Registre 38 :	DY7	DY6	DY5	DY4	DY3	DY2	DY1	DY0
Registre 39 :	0	0	0	0	0	0	DY9	DY8
Registre 40 :	NX7	NX6	NX5	NX4	NX3	NX2	NX1	NX0
Registre 41 :	0	0	0	0	0	0	0	NX8
Registre 42 :	NY7	NY6	NY5	NY4	NY3	NY2	NY1	NY0
Registre 43 :	0	0	0	0	0	0	NY9	NY8
Registre 44 :	CL7	CL6	CL5	CL4	CL3	CL2	CL1	CL0
Registre 45 :	0	MXC	MXD	MXS	DIY	DIX	EQ	MAJ
Registre 46 :	CM3	CM2	CM1	CM0	LO3	LO2	LO1	LO0

Liste alphabétique :

Nom	Registre	Fonction
AII	17	0 = Auto incrémentation ; 1 = Normal.
BC3 ~ BC0	12	En cas de clignotement ; couleur 2nde partie.
BD3 ~ BD0	7	« back drop color ».
BL	1	1 = Affichage autorisé ; 0 = Affichage interdit.
BW	8	1 = Noir et blanc (32 teintes) ; 0 = Couleur.
CB	8	1 = Bus de couleur en entrée ; 0 = En sortie.
CC3 ~ CC0	16	N° de couleur lors d'un accès palette.
CL3 ~ CL0	44	Couleur lors d'une commande.
CM3 ~ CM0	46	Code de commande à exécuter.
C13 ~ C6	3	Bits de poids faible qui définissent l'adresse de la table des couleurs.
C16 ~ C14	10	Bits de poids fort qui définissent l'adresse de la table des couleurs.
DC	9	1 = DTCLK en entrée ; 0 = DTCLK en sortie
DG	0	1 = Bus de couleur en entrée ; 0 = En sortie
DIX	45	Direction horizontale 1 = Gauche ; 0 = Droite
DIY	45	Direction verticale 1 = Haut ; 0 = bas
DO7 ~ DO0	23	Décalage vertical de l'écran (0-255)
DX7 ~ DX0	36	Abscisse destination (8 bits poids faible)
DX8	37	Abscisse destination (bit poids fort)
DY7 ~ DY0	38	Ordonnée destination (8 bits poids faible)
DY9 et DY8	39	Ordonnée destination (2 bits poids fort)
EQ	45	Srch : 1 = Couleur de bordure trouvée ; 0 = Couleur autre que bordure
E0	9	1 = Affichage alterné de 2 pages ; 0 = Normal

F16 ~ F11	4	Adresse table des formes (6 bits poids forts sur 17)
H3 ~ H0	18	Ajustement horizontal de l'écran (7 = Gauche ; 0 = Centre ; 8 = Droite)
IE0	1	1 = Interruption scan horizontal ok ; 0 = Interdit
IE1	0	1 = Interruption scan horizontal ok ; 0 = Interdit
IE2	0	1 = Interruption stylo optique ok ; 0 = Interdit
IL	9	1 = Affichage entrelacé ; 0 = Non entrelacé
IL7 ~ IL0	19	N° de ligne où l'interruption doit se déclencher lors d'un scan
LN	9	Résolution verticale de l'écran. 1 = 212 points ; 0 = 192 points
LO3 ~ LO0	46	Opérateur logique lors d'une commande
LP	8	1 = Stylo optique autorisé ; 0 = Stylo interdit
MAG	1	1 = Sprites agrandis ; 0 = Sprites normaux
MAJ	45	Côté le plus long 1 = Vertical ; 0 = Horizontal
MS	8	1 = Souris autorisée ; 0 = Souris interdite
MXC	45	Inutilisé sur msx2
MXD	45	Destination : 1 = Vram étendue/ 0 = Vram
MXS	45	Source : 1 = Vram étendue/ 0 = Vram
M2 ~ M1	1	2 bits de poids faible du mode d'écran
M5 ~ M3	0	3 bits de poids fort du mode d'écran
NT	9	1 = Pal (313 lignes) ; 0 = Ntsc (262 lignes)
NX7 ~ NX0	40	Longueur (8 bits poids faible)
NX8	41	Longueur (bit poids fort)
NY7 ~ NY0	42	Hauteur (8 bits poids faible)
NY9 et NY8	43	Hauteur (2 bits poids fort)
N16 ~ N10	2	Adresse table des caractères ou Bitmap (7 bits de poids fort sur 17)
OF3 ~ OF0	13	En cas de clignotement ; temps éteint
ON3 ~ ON0	13	En cas de clignotement ; temps allumé
P16 ~ P11	6	Adresse table génératrice des Sprites (6 bits de poids fort sur 17)
RS5 ~ RS0	17	N° de registre lors d'un adressage indirect
SI	1	1 = Sprites 16*16 ; 0 = Sprites 8*8
SPD	8	1 = Sprites interdits ; 0 = Sprites autorisés
ST3 ~ ST0	15	N° registre statut lors d'une lecture
SX7 ~ SX0	32	Abscisse source (8 bits poids faible)
SX8	33	Abscisse source (bit poids fort)
SY7 ~ SY0	34	Ordonnée source (8 bits poids faible)
SY9 et SY8	35	Ordonnée source (2 bits poids fort)
S1 et S0	9	Choix du mode simultané ou pas
S14 ~ S7	6	Adresse table des attributs des Sprites (10 bits...

S16 et S15	11	... de poids fort sur 17)
TC3 ~ TC0	7	Couleur du texte en screen 0 et 1
TP	8	Couleur 0 égale à la couleur de la palette
T23 ~ T20	12	En cas de clignotement, couleur 1ère partie
V3 ~ V0	18	Ajustement vertical de l'écran (8 = Bas ; 0 = milieu ; 7 = Haut)
V16 ~ V14	14	3 bits de poids fort de l'adresse VRAM
VR	8	Type de VRAM 1 = 64x1 bit ou 64x4 bits ; 0 = 16x1 bit ou 16x4 bits

E - Les cartes de la mémoire vidéo

La carte mémoire vidéo par défaut pour chaque mode graphique :

SCREEN 0 40 colonnes MSX1 à MSX turbo R

Table des caractères :	00000h~003BFh	960 octets
Table des formes :	00800h~00FFFh	2048 octets
Table de la palette (MSX2~) :	00400h~0041Fh	32 octets

SCREEN 0 80 colonnes MSX2 à MSX turbo R

Table des caractères 24 / 26,5 lignes :	00000h~0077Fh / 0086Fh	1920 / 2160 octets
Table des couleurs 24 / 26,5 lignes :	00800h~008EFh / 0090Dh	240 / 270 octets
Table des formes :	01000h~017FFh	2048 octets
Table de la palette (MSX2~) :	00F00h~00F1Fh	32 octets

SCREEN 1 32 colonnes MSX1 à MSX turbo R

Table des caractères :	01800h~01AFFh	768 octets
Table des couleurs :	02000h~0201Fh	32 octets
Table des formes :	00000h~007FFh	2048 octets
Table des attributs Sprite :	01B00h~01B7Fh	128 octets
Table des formes de Sprite :	03800h~03FFFh	2048 octets
Table de la palette (MSX2~) :	02020h~0203Fh	32 octets

SCREEN 2 MSX1 à MSX turbo R

table des motifs :	01800h~01AFFh	768 octets
Table des couleurs :	02000h~037FFh	6144 octets
Table des formes :	00000h~017FFh	6144 octets
Table des attributs Sprite :	01B00h~01B7Fh	128 octets
Table des formes de Sprite :	03800h~03FFFh	2048 octets
Table de la palette (MSX2~) :	02020h~0203Fh	32 octets

SCREEN 3 MSX1 à MSX turbo R

Table des caractères :	00800h~00AFFh	768 octets
Table des formes :	00000h~007FFh	2048 octets
Table des attributs Sprite :	01B00h~01B7Fh	128 octets
Table des formes de Sprite :	03800h~03FFFh	2048 octets
Table de la palette (MSX2~) :	02020h~0203Fh	32 octets

SCREEN 4 MSX2 à MSX turbo R

table des motifs :	01800h~01AFFh	768 octets
Table des couleurs :	02000h~037FFh	6144 octets
Table des formes :	00000h~017FFh	6144 octets
Table des attributs Sprite :	01E00h~01E7Fh	128 octets
Table des formes de Sprite :	03800h~03FFFh	2048 octets
Table des couleurs de Sprite :	01C00h~0D7FFh	512 octets
Table de la palette des couleurs :	01E80h~01E9Fh	32 octets

SCREEN 5 MSX2 à MSX turbo R

Table Bitmap 192 / 212 lignes :	00000h~05FFFh / 069FFh	24576 / 27136 octets
Table des attributs Sprite :	07600h~0767Fh	128 octets
Table des formes de Sprite :	07800h~07FFFh	2048 octets
Table des couleurs de Sprite :	07400h~075FFh	512 octets
Table de la palette des couleurs :	07680h~0769Fh	32 octets

SCREEN 6 MSX2 à MSX turbo R

Table Bitmap 192 / 212 lignes :	00000h~05FFFh / 069FFh	24576 / 27136 octets
Table des attributs Sprite :	07600h~0767Fh	128 octets
Table des formes de Sprite :	07800h~07FFFh	2048 octets
Table des couleurs de Sprite :	07400h~075FFh	512 octets
Table de la palette des couleurs :	07680h~0769Fh	32 octets

SCREEN 7 MSX2 à MSX turbo R

Table Bitmap 192 / 212 lignes :	00000h~0BFFFh / 0D3FFh	49152 / 54272 octets
Table des attributs Sprite :	0FA00h~0FA7Fh	128 octets

Table des formes de Sprite :	0F000h~0F7FFh	2048 octets
Table des couleurs de Sprite :	0F800h~0F9FFh	512 octets
Table de la palette des couleurs :	0FA80h~0FA9Fh	32 octets

SCREEN 8 MSX2 à MSX turbo R

Table Bitmap 192 / 212 lignes :	00000h~0BFFFh / 0D3FFh	49152 / 54272 octets
Table des attributs Sprite :	0FA00h~0FA7Fh	128 octets
Table des formes de Sprite :	0F000h~0F7FFh	2048 octets
Table des couleurs de Sprite :	0F800h~0F9FFh	512 octets
Table de la palette des couleurs :	0FA80h~0FA9Fh	32 octets

SCREEN 9 MSX2 coréen

Table Bitmap 192 / 212 lignes :	00000h~05FFFh / 069FFh	24576 / 27136 octets
Table des attributs Sprite :	00800h~0767Fh (<i>inutilisé par le Basic</i>)	128 octets
Table des formes de Sprite :	07800h~07FFFh (<i>inutilisé par le Basic</i>)	2048 octets
Table des couleur des Sprite :	07400h~075FFh (<i>inutilisé par le Basic</i>)	512 octets
Table de la palette des couleurs :	07680h~0769Fh	32 octets

SCREEN 10 / 11 MSX2+

Table Bitmap 192 / 212 lignes :	00000h~0BFFFh / 0D3FFh	49152 / 54272 octets
Table des attributs Sprite :	0FA00h~0FA7Fh	128 octets
Table des formes de Sprite :	0F000h~0F7FFh	2048 octets
Table des couleurs de Sprite :	0F800h~0F9FFh	512 octets
Table de la palette des couleurs :	0FA80h~0FA9Fh	32 octets

SCREEN 12 MSX2+

Table Bitmap 192 / 212 lignes :	00000h~0BFFFh / 0D3FFh	49152 / 54272 octets
Table des attributs Sprite :	0FA00h~0FA7Fh	128 octets
Table des formes de Sprite :	0F000h~0F7FFh	2048 octets
Table des couleurs des Sprites :	0F800h~0F9FFh	512 octets
Table de la palette des couleurs :	0FA80h~0FA9Fh	32 octets

F - Jeux de caractères MSX

Les jeux de caractères MSX sont tous basés sur le code ASCII 7 bits et étendu de la façon suivante sur les MSX occidentaux / internationaux :

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NULL	GRAPH	CTRL B	CTRL C	CTRL D	CTRL E	CTRL F	BEEP	BS	TAB	LF	HOME	CLS	RET	CTRL N	CTRL O
1	CTRL P	CTRL Q	INS	CTRL S	CTRL T	CTRL U	CTRL V	CTRL W	SEL	CTRL Y	CTRL Z	ESC	→	←	↑	↓
2	SPC	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL
8	ç	ü	é	à	ä	å	ç	ê	ë	è	ï	î	ï	Ä	Å	
9	É	æ	Æ	ø	ö	õ	ù	û	ÿ	Ö	Ü	ç	€	¥	₣	f
A	á	í	ó	ú	ñ	ñ	ä	ö	ç	ı	ı	½	¼	ı	«	»
B	ä	ä	ï	ï	ö	ö	ü	ü	ı	ı	ı	ı	ı	ı	ı	ı
C	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
D	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
E	α	β	γ	π	Σ	σ	μ	γ	ϕ	θ	Ω	δ	ω	ø	€	π
F	≡	±	≥	≤	↑	↓	÷	×	○	•	.	√	°	²	■	■

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4		☺	☹	♥	♣	♠	♣	•	■	○	☉	♂	♀	♫	♫	✳
5	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

- Les 8 premières lignes sont les mêmes caractères que ceux du code ASCII 7 bits.
- Les lignes 0 et 1 ainsi que la case 6Fh sont des codes de contrôle qui ne s'affichent pas à l'écran mais ont pour effet suivant.

GRAPH = Permet d'afficher un caractère graphique étendu.

CTRL B = Place le curseur sur le mot précédent.

CTRL E = Efface la ligne à droite du curseur.

CTRL F = Place le curseur sur le mot suivant.

LF = Descend le curseur d'une ligne.

HOME = Positionnement du curseur tout en haut à gauche.

CTRL N = Place le curseur en fin de ligne.

INS = Passage en mode insertion / écraser.

CTRL U = Efface la ligne où se trouve le curseur.

SEL = Touche de sélection nom.

↑ = Curseur vers le haut.

→ = Curseur vers la droite.

← = Curseur vers la gauche.

↓ = Curseur vers le bas.

- Les caractères verts sont des caractères spécifiques aux MSX occidentaux.
- Le dernier caractère (FFh) correspond au curseur.
- La table du bas, à deux lignes (orange), sont des caractères graphiques étendus. Pour les afficher, chacun doit être précédé du caractère 01h. Ils sont aussi spécifiques au MSX occidentaux.

Le jeu de caractères des MSX japonais :

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NULL	GRPH	CTRL B	CTRL C	CTRL D	CTRL E	CTRL F	BEEP	BS	TAB	LF	HOME	CLS	RET	CTRL N	CTRL O
1	CTRL P	CTRL R	INS	CTRL S	CTRL T	CTRL U	CTRL V	CTRL W	SEL	CTRL Y	CTRL Z	ESC	→	←	↑	↓
2	SPC	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[¥]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL
8	▲	■	◆	●	○	●	を	あ	い	う	え	お	や	ゆ	よ	っ
9		あ	い	う	え	お	か	き	く	け	こ	さ	し	す	せ	ぞ
A		。」「	、	・	ヲ	ア	イ	ウ	エ	オ	カ	ユ	ヨ	ツ		
B	ー	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ
C	ヲ	チ	ツ	テ	ト	ナ	ニ	ヌ	ノ	ハ	ヒ	フ	ヘ	ホ	マ	
D	ミ	ム	メ	モ	カ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ン	〃	。
E	た	ち	つ	て	と	な	に	ぬ	の	ほ	ひ	ふ	へ	ほ	ま	
F	み	む	め	も	や	ゆ	よ	ら	り	る	れ	ろ	わ	ん		■

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4		月	火	水	木	金	土	日	年	円	時	分	秒	百	千	万
5	元	十	十	十	十	十	十	一	一	一	一	一	一	一	一	一

- Les 8 premières lignes sont les mêmes caractères que ceux du code ASCII 7 bits. Excepté le caractère « \ » (5Ch) qui a été remplacé par le caractère « ¥ » du code JIS 8 bits de Microsoft. (Le code JIS 8 bits est basé sur le code ASCII 7 bits.)

- Les caractères violets proviennent du code JIS 8 bits.
- Les caractères verts, bien que non définis par la norme JIS et laissé au libre arbitre des , sont JIS 8 bits les mêmes Hiragana.
- Le dernier caractère (FFh) correspond au curseur.
- La table du bas, à deux lignes, sont des caractères étendus propres au MSX. Pour les afficher, chacun doit être précédé du caractère 01h.

Notes :

Les MSX2, ou plus récents, japonais possèdent en option ou en interne une Kanji ROM comprenant un police de caractères au format JIS 16 bits niveau 1 et niveau 2 pour les MSX plus récents.

Ces MSX ou cartouches ont en général aussi un « Kanji BASIC » et un « MSX-JE » pour les MSX plus récents. Le « Kanji BASIC » ajoute au MSX la possibilité d'afficher les caractères japonais, les Kanji et même des modes d'écran de texte adaptés aux Kanji. Le « MSX-JE » sert à faciliter la saisie des Kanji.

Le jeu de caractères des MSX arabes :

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NULL	GRPH	CTRL B	CTRL C	CTRL D	CTRL E	CTRL F	BEEP	BS	TAB	LF	HOME	CLS	RET	CTRL N	CTRL O
1	CTRL P	CTRL R	INS	CTRL S	CTRL T	CTRL U	CTRL V	CTRL W	SEL	CTRL Y	CTRL Z	ESC	→	←	↑	↓
2	SPC	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL
8	é	á	à	ç	ê	ë	è	ï	î	ó	ô	ù	õ	ö	no	no
9	p	q	r	s	t	u	v	w	x	y	z	{		}		
A		!	"	#	\$	%	&	')	(*	+	,	-	.	/
B	+	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
C	@	ء	آ	إ	ق	ك	ل	ب	ة	ث	ج	ح	خ	د		
D	ذ	ر	ز	س	ش	ص	ض	ط	ظ	ع	غ	ف	ق	ك	ل	م
E	ن	ف	ظ	ك	ل	م	ن	هـ	و	ي	پ	ی	”	”	’	’
F	،	”	”	”	”	”	”	”	”	”	”	”	”	”	”	”

- Les 8 premières lignes sont les mêmes caractères que ceux du code ASCII 7 bits.
- Les caractères verts sont des caractères propres aux MSX arabes.
- Les MSX arabes n'ont pas de caractères étendus.

Note :

Les MSX arabes ont deux Main-ROM, lorsqu'on presse la touche CTRL pendant le démarrage du MSX jusqu'au son « bip », le MSX démarre sur la Main-ROM ayant un jeu de caractères international.

Le jeu de caractères des MSX coréens :

Les MSX coréens utilisent une police de caractères basée sur le code ASCII 7 bit avec les caractères coréens en plus.

Les 128 premiers caractères sont les mêmes caractères que ceux du code ASCII. Excepté le caractère « \ » (5Ch) qui a été remplacé par le caractère « ₩ ».

Les caractères suivants s'affichent d'une façon assez spécifique. Un caractère coréen (Hangul) est composé de 2 ou 3 caractères mais prend la place de quatre caractères 8x8 (deux au dessus deux en dessous) à l'écran.

Les MSX2 coréens, ont un mode d'écran spécifique (le SCREEN 9). Ce mode est un mode texte spécifique adapté au coréen basé sur le SCREEN 6. Ces MSX2 sont équipés d'une ROM supplémentaire de 32Ko pour afficher les caractères Hangul.

G - Exemples de matrices de clavier

Le caractère à droite correspond à celui obtenu avec la touche SHIFT maintenue.

Clavier « AZERTY » du Philips NMS-8250 et Sony HB-F700 :

Ligne	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
00	è 7	§ 6	(5	' 4	" 3	é 2	& 1	à 0
01	m M	\$ *	^ "	< >	- _) °	ç 9	! 8
02	b B	q Q		= +	: /	; .	# £	ù %
03	j J	i I	h H	g G	f F	e E	d D	c C
04	r R	a A	p P	o O	n N	, ?	l L	k K
05	w W	x X	y Y	z Z	v V	u U	t T	s S
06	F3 F8	F2 F7	F1 F6	CODE	CAPS	GRAPH	CTRL	SHIFT
07	RET	SELECT	BS	STOP	TAB	ESC	F5 F10	F4 F9
08	→	↓	↑	←	SUP	INS	Home EFF	SPACE
09	4	3	2	1	0	/	+	*
10	.	,	-	9	8	7	6	5

Les deux dernières lignes sont celles du pavé numérique.

Clavier « AZERTY » du Canon V-20, Sanyo PHC-28 et Yeno MX-64 :

Ligne	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
00	7 è	6 §	5 (4 '	3 "	2 é	1 &	0 à
01	m M	\$ *	^ "	< >	- _) °	9 ç	8 !
02	b B	q Q	DEAD	= +	: /	; .	# £	ù %
03	j J	i I	h H	g G	f F	e E	d D	c C
04	r R	a A	p P	o O	n N	, ?	l L	k K
05	w W	x X	y Y	z Z	v V	u U	t T	s S
06	F3 F8	F2 F7	F1 F6	CODE	CAPS	GRAPH	CTRL	SHIFT
07	RET	SELECT	BS	STOP	TAB	ESC	F5 F10	F4 F9
08	→	↓	↑	←	SUP	INS	Home EFF	SPACE

Touche « DEAD » : Sanyo PHC-28.

Clavier « AZERTY » du Philips VG-8020, Sanyo PHC-28L, Schneider MC-810 et Yeno DPC-64 :

Ligne	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
00	è 7	§ 6	(5	' 4	" 3	é 2	& 1	à 0
01	m M	\$ *	^ "	< >	- _) °	ç 9	! 8
02	b B	q Q	DEAD	= +	: /	; .	# £	ù %
03	j J	i I	h H	g G	f F	e E	d D	c C
04	r R	a A	p P	o O	n N	, ?	l L	k K
05	w W	x X	y Y	z Z	v V	u U	t T	s S
06	F3 F8	F2 F7	F1 F6	CODE	CAPS	GRAPH	CTRL	SHIFT
07	RET	SELECT	BS	STOP	TAB	ESC	F5 F10	F4 F9
08	→	↓	↑	←	SUP	INS	Home EFF	SPACE

Touche « DEAD » : Sanyo PHC-28L.

Clavier « QWERTY » (Japonais) des Panasonic MSX2+ et MSX turbo R :

Ligne	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
00	7 ' 6 &	5 %	4 \$	3 #	2 "	1 !	0	
01	; +	[{	@ `	¥	^ ~	- =	9)	8 (
02	b B	a A	_	/ ?	. >	, <] }	: *
03	j J	i I	h H	g G	f F	e E	d D	c C
04	r R	q Q	p P	o O	n N	m M	l L	k K
05	z Z	y Y	x X	w W	v V	u U	t T	s S
06	F3 F8	F2 F7	F1 F6	かな	CAPS	GRAPH	CTRL	SHIFT
07	RET	SELECT	BS	STOP	TAB	ESC	F5 F10	F4 F9
08	→	↓	↑	←	SUP	INS	Home CLS	SPACE
09	4	3	2	1	0	/	+	*
10	.	,	-	9	8	7	6	5
11					取消		実行	

Les deux dernières lignes sont celles du pavé numérique.

Clavier russe du Yamaha KYBT1 YIS-503II et KYBT2 YIS-503IIIR :

Ligne	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
00	& 6	% 5	α 4	# 3	" 2	! 1	+ ;) 9
01	V Ж	* :	H X	- ^ Ъ	= _	\$ 0	(8	' 7
02	I И	F Ф	? /	< ,	@ Ю	Б Б	> .	\ Э
03	o O	[{ Ш	R P	P П	A A	U y	W B	S C
04	K К	J Ё	Z 3] } Щ	T T	X Ь	D Д	L Л
05	Q Я	N H	! ~ Ч	C Ц	M M	G Г	E E	Y Ы
06	F3 F8	F2 F7	F1 F6	РУС	CAPS	GRAPH	CTRL	SHIFT
07	RET	SELECT	BS	STOP	TAB	ESC	F5 F10	F4 F9
08	→	↓	↑	←	SUP	INS	Home CLS	SPACE

Clavier russe du Yamaha KYBT2 YIS-805 :

Ligne	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
00	& 6	% 5	␣ 4	# 3	" 2	! 1	+ ;) 9
01	V Ж	* :	H X	- ^ Ъ	= _	\$ 0	(8	' 7
02	I И	F Ф	? /	< ,	@ Ю	B Б	> .	\ Э
03	o O	[{ Ш	R P	P П	A A	U y	W B	S C
04	K К	J Й	Z 3] } Щ	T T	X Ъ	D Д	L Л
05	Q Я	N H	~ Ч	C Ц	M M	G Г	E E	Y Ы
06	F3 F8	F2 F7	F1 F6	РУС	CAPS	GRAPH	CTRL	SHIFT
07	RET	SELECT	BS	STOP	TAB	ESC	F5 F10	F4 F9
08	→	↓	↑	←	SUP	INS	Home CLS	SPACE
09	4	3	2	1	0	RET	+	*
10	.	,	-	9	8	7	6	5

Clavier russe du Sony HB-G9P :

Ligne	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
00	& 6	% 5	␣ 4	# 3	" 2	! 1	+ ;) 9
01	V Ж	* :	H X	- ^ Ъ	= _	\$ 0	(8	' 7
02	I И	F Ф	? /	< ,	@ Ю	B Б	> .	\ Э
03	o O	[Ш	R P	P П	A A	U y	W B	S C
04	K К	J Й	Z 3] Щ	T T	X Ъ	D Д	L Л
05	Q Я	N H	Ч	C Ц	M M	G Г	E E	Y Ы
06	F3 F8	F2 F7	F1 F6	Code	CAPS	GRAPH	CTRL	SHIFT
07	RET	SELECT	BS	STOP	TAB	ESC	F5 F10	F4 F9
08	→	↓	↑	←	SUP	INS	Home CLS	SPACE
09	4	3	2	1	0	/	+	*
10	.	,	-	9	8	7	6	5

H - Codes d'erreur du Basic et du disque Basic

Code	Message	Description
1	NEXT without FOR	L'interpréteur a rencontré une instruction NEXT sans le FOR au préalable.
2	Syntax error	Une instruction a une mauvaise syntaxe.
3	RETURN without GOSUB	L'interpréteur a rencontré une instruction RETURN sans le GOSUB au préalable.
4	Out of DATA	READ a été exécuté alors qu'il n'y a plus de donnée à lire dans DATA.
5	Illegal function call	Une valeur dépasse la limite possible dans la fonction.
6	Overflow	La valeur d'une variable dépasse la limite possible.
7	Out of memory	La mémoire allouée au Basic est pleine.
8	Undefined line number	Une instruction indique une ligne inexistante.
9	Subscript out of range	Les paramètres d'une variable dépassent la taille du tableau.
10	Redimensioned array	Le tableau a déjà été créé.
11	Division by zero	On ne peut pas diviser par zéro.
12	Illegal direct	L'instruction entrée en mode direct ne peut être exécutée qu'en mode programme.
13	Type mismatch	Une valeur numérique a été affectée en tant que chaîne alpha-numérique ou vice-versa.
14	Out of string space	
15	String too long	La chaîne alpha-numérique dépasse les 256 caractères.
16	String formula too complex	La chaîne alpha-numérique est composée de trop de fonctions.
17	Can't CONTINUE	Le programme ne peut continuer son exécution.
18	Undefined user function	
19	Device I/O error	
20	Verify error	
21	No RESUME	La routine de traitement d'erreur doit se terminer par RESUME.
22	RESUME without error	RESUME a été exécutée alors qu'il n'y a pas eu d'erreur.
23	Unprintable error	Cette erreur n'a pas de message.
24	Missing operand	L'opérateur d'une expression n'a pas d'opérande.
25	Line buffer overflow	La ligne de programme entrée est trop longue. (256 caractères max.)
26-49	Indéfini	
50	FIELD overflow	Le nombre de caractères de FIELD dépassent les 256.
51	Internal error	Une erreur s'est produite dans le système.
52	Bad file number	Mauvais numéro de fichier.
53	File not found	Le fichier n'a pas été trouvé.
54	File already open	Le fichier que l'on veut ouvrir (ou effacer) est déjà ouvert.
55	Input past end	INPUT# a tenté de d'entrer une donnée hors du fichier.
56	Bad file name	Mauvais nom de fichier. Certains caractères ne sont pas utilisables.
57	Direct statement in file	

58	Sequential I/O only	On essaie de lire un fichier séquentiel par accès direct.
59	File not OPEN	Le fichier n'a pas été ouvert.
60	Bad Allocation Table	La table allouée aux fichiers (FAT) est endommagée.
61	Bad file mode	Mauvaise utilisation de fichier. PUT ou GET a été utilisé sur un fichier séquentiel ou l'ouverture d'un fichier d'un format qui ne correspond pas a été tenté.
62	Bad drive name	Le nom de lecteur employé est différent de « A: », « B: », ... ou « H: ».
63	Bad sector number	
64	File still open	Un fichier est encore ouvert.
65	File already exists	Il y a déjà un fichier ayant le même nom au même endroit sur le disque.
66	Disk full	Il n'y a plus d'espace libre sur le disque.
67	Too many files	Le nombre de fichiers excède celui défini par MAXFILES.
68	Disk write protected	Une tentative d'écriture a été faite sur un disque protégé contre l'écriture.
69	Disk I/O error	Le système a trouvé une erreur lors de la lecture ou l'écriture sur le disque.
70	Disk offline	
71	Rename across disk	
72~255	Indéfini	

Voir un guide du MSX-BASIC pour plus de précision.